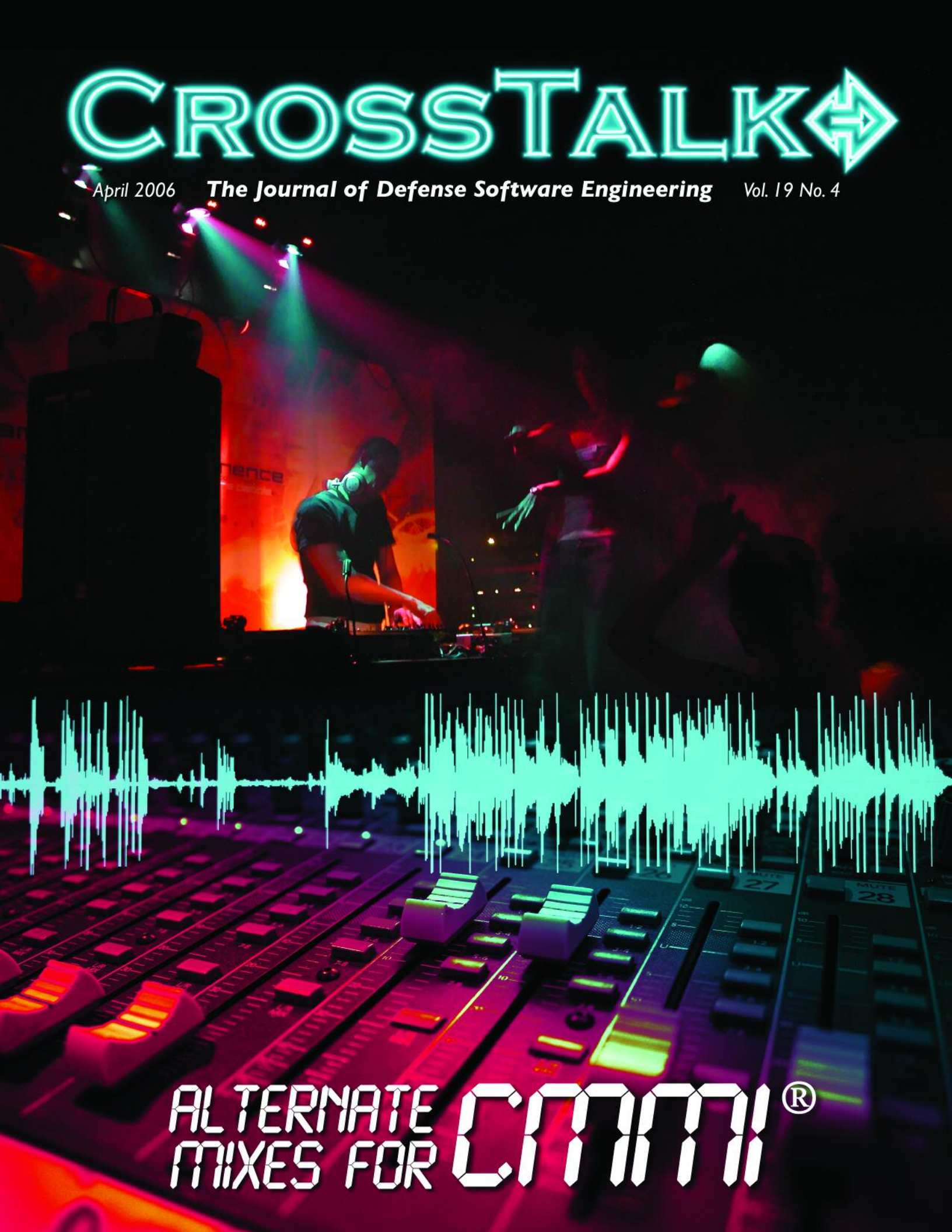
CrossTalk

April 2006 The Journal of Defense Software Engineering Vol. 19 No. 4



Alternate Mixes for CMMI®

Alternate Mixes for CMMI

Software Engineering Technology

Open Forum

On The Cover
Cover Design by Kent Bingham

Additional art services provided by Janna Jensen. jensendesigns@aol.com

Departments

CrossTalk

76 SMXG Co-Sponsor	*Kevin Stamey*
309 SMXG Co-Sponsor	*Randy Hill*
402 SMXG Co-Sponsor	*Bob Zwitch*
DHS Co-Sponsor	*Joe Jarzombek*
NAVAIR Co-Sponsor	*Jeff Schwalb*
Publisher	*Brent Baxter*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Pamela Palmer*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Phone	(801) 775-5555
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Air Force (USAF), the U.S. Department of Homeland Security (DHS), and the U.S. Navy (USN). USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG), Ogden-ALC 309 SMXG, and Warner Robins-ALC 402 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection. USN co-sponsor: Naval Air Systems Command (NAVAIR) Software Systems Support Center.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 18.

309 SMXG/MXDB
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, or the STSC. All product names referenced in this issue are trademarks of their companies.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Transitioning to a New Model? First Consider Your Organizational Culture

The decision to transition my organization from the Capability Maturity Model® for Software (SW-CMM®) to CMM Integration (CMMI®) was not an easy one. We had spent the previous decade immersed in change, growing from an ad-hoc organization to becoming the first Department of Defense software engineering house to achieve full SW-CMM Level 5 compliance. It has been a rewarding and arduous journey. However, the advent of CMMI led us to the proverbial crossroads in the path we were traveling: Do we plunge in and incorporate the additional CMMI practices into our organizational processes and in so doing stay at the leading edge of process maturity? Or do we simply utilize the robust software process improvement infrastructure we had developed to incrementally get better and better over time? The last question was actually a compelling argument against transitioning to the CMMI model. After all, the ability to affect continuous process improvement was the reason we adopted the Level 4 and 5 key practices of the SW-CMM in the first place.

In the end, the concern over the planned obsolescence of the SW-CMM model and the additional benefits of CMMI swung the debate in favor of the transition. I knew it was the right thing to do, but nonetheless had apprehensions about leading our team in that direction. How would this decision be received by my senior staff and our sizeable work force? I feared it could be viewed as more change for change's sake, or simply as an attempt to keep up with the latest management fad. If handled incorrectly, I could create an entire organization of process improvement cynics!

I knew that I needed to communicate our goals and objectives to every employee. So I visited with each unit and explained how we had benefited from the SW-CMM, why we needed to transition to CMMI, how I expected CMMI would benefit us, how we planned to make the transition, and how the transition would affect each of them as individuals. I can't say that everyone was excited about making the change, but everyone understood why we were doing it; they supported it and made it happen. More importantly, I discovered that we had developed something very special over the previous 10 years of incorporating model-based process improvement: We institutionalized an evolutionary change of sorts across our organization, developing the modern-day survival equivalent of the chameleon's ability to adapt to changing environments. We had created an organizational culture that understands and accepts change. That may ultimately prove to be one of the greatest values to us that have resulted from our SW-CMM, and now CMMI, compliance.

Dr. Kenneth Shere observes in his article, *Managing Cultural Changes in Your Organization,* that process improvement causes cultural change, change is hard, and that one must have a compelling reason for change. I encourage you to take advantage of his recommendations when embarking on your next organizational change adventure.

Another recent change being accepted by my organization affects CrossTalk as I move Tracy Stauder from the CrossTalk publisher position into a supervisory position. Tracy has supported CrossTalk since 1996 when she took the lead in its production. In cooperation with our organization's desire to implement consistent CMM processes, Tracy formalized and documented CrossTalk's production processes. Those who have been subscribing to CrossTalk since before she joined might have noticed that as a result of her leadership, CrossTalk's production became much more predictable and the quality also improved.

These improvements reflect Tracy's outstanding contributions to my organization. Her leadership talents will benefit her new group, and this opportunity will reward her excellent work. One basic concept of process maturity is that by having established processes in place, an organization will not depend on one hero to keep it running. Tracy was a hero, but was also true to our process improvement initiatives, which will ensure CrossTalk's continued quality in this evolutionary change.

Randy B. Hill
Ogden Air Logistics Center, Co-Sponsor

Army Simulation Program Balances Agile and Traditional Methods With Success

LTC John Surdu, Ph.D.
U.S. Army

Doug J. Parsons
Program Executive Office – Simulation Training and Instrumentation

The One Semi-Automated Forces (OneSAF) Objective System is the next generation simulation system planned to provide the U.S. Army with an entity-level simulation to serve three modeling and simulation domains. Software development of the OneSAF application has been conducted in a highly robust systems engineering environment based on commercial and government best practices. The OneSAF program has tailored techniques of Extreme Programming (XP) and other agile methods into a development environment that has resulted in several industry awards, most recently the National Training Systems Association Cross Function Award for the Integrated Product Team. These externally certified Capability Maturity Model® Integration Level 5 processes are credited with successful program execution. This article will discuss which XP and other agile techniques were used, which were not, and why.

Principles and practices associated with agile methods are not new concepts in the world of software development. Extreme Programming (XP), an agile method itself, has been used successfully in a variety of software development environments. Overall reviews in the software development community have been mixed. In his history of the Agile Manifesto [1], Jim Highsmith uses terms like *organizational anarchists* and *independent thinkers* to describe the alliance embracing a unifying set of values. The following is stated in the Agile Manifesto:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
> - Individuals and interactions over processes and tools.
> - Working software over comprehensive documentation.
> - Customer collaboration over contract negotiation.
> - Responding to change over following a plan.
>
> That is, while there is value in the items on the right, we value the items on the left more. [2]

Pure software development traditionalists consider the techniques that many agile software developers use to follow these values as an excuse to hack undocumented, poorly designed code. The pure agile developer would consider the plan-driven traditionalists responsible for saddling software developers with low-value processes that hinder or prevent the delivery of software. Fortunately, we, as software developers, do not need to take sides. There exist many shades of gray in the spectrum between agile and traditional methods. The shade you select that best fits your program depends upon many factors, including (but not limited to) team size, criticality of defects, ability to receive user feedback/interaction, customer expectations, budget, schedule, and stability of requirements. Since these factors are different from program to program, no two programs should have identical strategies in their software development.

"There exist many shades of gray in the spectrum between agile and traditional methods ... no two programs should have identical strategies in their software development."

Traditional methods are geared toward optimization, predictability, and control. Agile methods focus on adaptation to change, flexibility, and innovation. The new art of software development is finding the appropriate balance point among the available practices. Figure 1 is a chart developed by McCabe and Polen [3] based on a figure from Alistair Cockburn [4] indicating the relative nature between agile and conventional projects based on project team size and notional cost of system failure. Included on the chart is the placement of a U.S. Army simulation program, known as the One Semi-Automated Forces (OneSAF) Objective System (OOS).

The OOS is a large software-intensive system (greater than two million lines of source code) that will be used to support research and development and train future U.S. military leaders. In the genesis of the OOS program development, we researched and considered aspects of agile methods and XP as well as traditional strategies. None of these approaches were looked on as cookbook methodologies, but rather as a smorgasbord. Those that appeared to be a good fit were implemented; others were left on the table.

After more than four years of development, the OneSAF program remains on-track to satisfy requirements and meet user needs. The OneSAF program has been awarded the National Training Systems Association Cross Function Award for the Integrated Product Team; it was also selected by CROSSTALK as one of the 2003 U.S. Government's Top 5 Quality Software Projects.

Interestingly enough, the customer base for OOS was more comfortable with *traditional* pedantic software-development methods, even though they contributed to an environment with ill-defined and often-changing requirements. The program spent a great deal of time educating the user representatives on XP and other agile methods. Most have grown to embrace these methods, but the program still hears some users making statements like, "Spiral development doesn't work; it has created a configuration management nightmare," or "XP has reduced time for testing." Despite statements like these, the processes established for OOS development – a blend of agile, extreme, and traditional techniques – have been instrumental in the program reaching programmatic and technical goals.

This article is not intended to discuss the technical capabilities of the OOS;

® Capability Maturity Model is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

however, it is important that the reader understands some of the program's history to note similarities and distinctions with their own programs, present or past.

Mission Need and the Users

The OOS is the Army's next generation simulation system that can represent a full range of military operations, systems, and control processes. It is an entity-level simulation, meaning that it can simulate the activities of individual combatants or vehicles (as opposed to aggregate-level simulations, which represent combatants and vehicles as groupings). It will also provide the appropriate representations of the physical environment (e.g., terrain features, weather, and illumination) and its effect on simulated activities and behaviors.

The OOS is unique among Army simulations in that it is designed for use by three distinct Army Modeling and Simulation (M&S) domains. Specifically, the Advanced Concepts and Requirements domain uses M&S for experimentation and analysis on Army doctrine and force-related concepts. The Research, Development, and Acquisition domain uses M&S for acquisition analyses focused on equipping and supporting currently fielded and future forces. Finally, the Training, Exercises, and Military Operations domain employs M&S to train forces. It does so using live simulation (actual equipment on training ranges), virtual simulation (immersing the trainee into a synthetic environment), and constructive simulation (war games using computer-generated forces).

There exist two factors relative to our users that inherently pull the OOS development toward the center of the agile-conventional project spectrum. First, OOS will be used extensively for analysis and experimentation. Analysts and research scientists rely on a robust set of documentation to support verification and validation (V&V). For these users, it is critical to understand how OOS models work. Secondly, the OneSAF business model includes the distribution of source code with release of the software baseline [5]. The intention is to create an environment that will optimize the ability for extended capabilities created by the M&S community to be reintegrated into the baseline. Agile projects tend to be light on documentation and process. OOS will include a robust set of documentation and tools to support V&V. In addition, process description and documentation to aid and guide external developers of the baseline will also be provided.

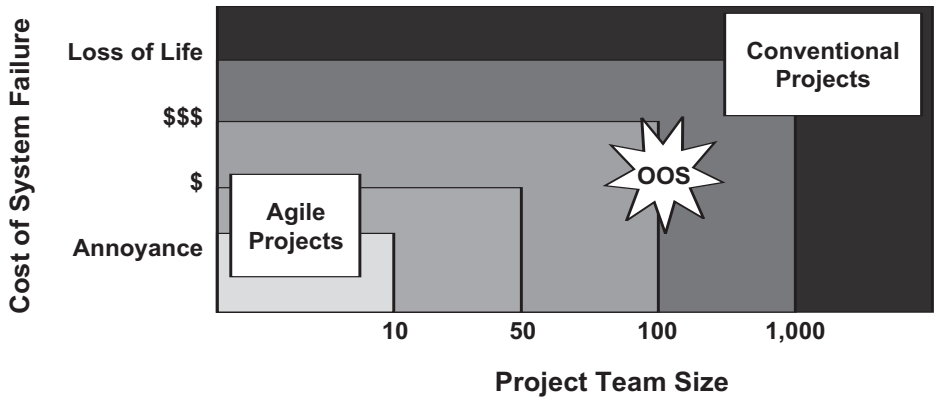


Figure 1: *Spectrum for Project Type Considering Defect Cost and Team Size*

A New Approach

While the use of commercial best practices seems intuitive, five years ago this was quite unique in Department of Defense (DoD) software development. Some of the changes were simple. The approach toward programmatic documentation was a minimalist one. Some approaches were considered heretical: utilizing the OneSAF program manager (PM) as the manager of the OOS task orders (similar to a *prime contractor*), and establishing an integrated development environment (IDE), which made the adoption of many agile methods and XP practices possible.

PM Is the Prime

A typical approach in DoD software development is for the government to select a single contractor who specifies its own set of subcontractors under a large, monolithic contract. In contrast, PM OneSAF was allowed to complete a variety of task orders under an Indefinite Delivery, Indefinite Quantity contract and manage it as the lead systems integrator. The advantage is that the government can pick a *best-of-breed* contractor rather than settle for a sub-contractor who may or may not be the best choice. During the past four years, 26 different contractor companies have been contracted to work on different parts of the OneSAF software. The contracts range in scope and include short-term studies, architecture review, knowledge acquisition, architecture and integration, model development, and integration and test.

An initial concern was that the contractors would disagree on the means to resolve issues and the process would grind to a halt. Associate Contractor Agreements (ACAs) were signed by each task order organization; however, an ACA is only a piece of paper that asks a contractor to play nice. To help *socialize* the ACAs, the contracts were awarded over the space of 18 months. The Architecture and Integration (A&I) contract was awarded first, and the processes, tools, and procedures were established. When new contractors came on board, they were integrated into existing processes so that we avoided *food fights* about whose processes were better than others. This does not imply that existing processes were never modified; we remained committed throughout execution to continuous process improvement and aggressively sought new ideas.

Three factors contributed to the ability for PM OneSAF to successfully act as the prime. First, the OneSAF government team is involved in the day-to-day development process. This allows informed and timely decisions to be made on behalf of the PM. Second, PM OneSAF empowered the A&I contractor with a great deal of flexibility to establish development, integration, and test processes from industry best practices: There were no government-mandated processes heavy with valueless documentation. Third, the PM sought technically qualified folks across the breadth of the program. Not only are OneSAF engineers truly technical with advanced degrees in software-related disciplines and years of real engineering experience, but OneSAF managers and project directors were recruited from engineering. As there are numerous disincentives for technical people within the Army, it was quite challenging to find even the small number of qualified engineers we needed.

It is unclear whether these unique business processes alone were the major contributor to successful execution or whether the employment of agile methods was the key. It is clear, however, that without these new ways of doing business, it is unlikely that agile methods would have been employed or embraced.

IDE

To support an effective and efficient soft-

ware development process, PM OneSAF established an environment that brought together domain/user representatives, government engineers, PMs, and contract or software developers. All of these teams are collocated in a single facility. A task order for facility operation and sustainment was intentionally awarded to a contractor outside any of those developing software to send the message that the IDE was *neutral turf*. Over the past four years, there have been as many as a dozen individual task orders under execution at any one time. From these task orders, there have been more than 100 software engineers working in concert to develop a baseline.

Applying Agile Methods and XP to OneSAF

Communication and Collocation

The Agile Manifesto states that in agile software development, the following should occur [6]:

- Business people and developers must work together daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Paulk states that agile methods generally apply to smaller teams working in the face of vague and/or dynamic requirements [7]. He also states that agile teams are expected to be collocated with typically fewer than 10 members. XP rules also indicate that the customer should always be available and that a stand-up meeting starts each day to communicate problems, solutions, and promote team focus.

OOS development has practiced collocation from its inception. As mentioned earlier, the government program office is the *prime* for OOS, and there are numerous contractors working the program. All contractors from the 10 to 12 companies working on OOS are collocated in the IDE facility. In addition, the combat developer (requirements steward and customer), customer representatives (from the three Army M&S domains), and government engineers and managers are collocated in the same facility. It often takes a newcomer to the program months to find out what company everyone works for because on Team OneSAF we concentrate on functional decomposition of the problem more than on which company is working a portion of the problem.

Through the use of ACAs, communication between the various organizations is smooth and seamless. The face-to-face coordination referred to in the Agile Manifesto occurs habitually in the IDE. A member of the combat development team said recently, "Much of the work on OOS occurs in the hallways." Informal meetings with the right two or three people in the hallway often work through some technical or inter-team coordination issue in a few minutes rather than scheduling a meeting with dozens of folks. OOS developers are encouraged to get up and walk down the hallway if they have an issue. From the earliest days of OOS development, the OneSAF PM Office prohibited weekly scheduled meetings on Wednesday through Friday. A lesson learned from other program development efforts was that engineers tended to save communication with other people on the team until the scheduled meeting. Intentionally, this forced the engineers to meet more frequently in desk-side or hallway meetings.

Spiral Development: Builds, Blocks, and Early Delivery

In discussing requirements, McCabe and Polen state:

> … the customer may hardly grasp the problem, much less the best system to address it. Therefore, requirements are likely to be vague or speculative when they should be specific. [3]

Recognizing that the customers' requirements and desires would evolve as they saw working prototypes (or alpha versions of the software), OOS adapted a spiral development methodology. Paulk noted:

> … with their emphasis on addressing requirements volatility, agile methodologies could be a powerful synthesis of practices that DoD contractors could leverage to make planning more responsive to change. [7]

The Agile Manifesto supports the notion of spiral development with the following tenets [6]:

- The highest priority is to satisfy the customer though early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Working software is the primary measure of progress.

XP rules and practices concerning the software development process include the following [8]:

- Make frequent small releases.
- The project is divided into iterations.
- Integrate often.
- Iteration planning starts each iteration.
- Never add functionality early.
- Code must be written to agreed-upon standards.

The development methodology adopted for OOS was one of frequent iterations, or spirals. It involved breaking the overall program into a series of eight- to 10-week builds. Several of these builds were then designated as user assessment baselines that were made available to users for assessment and azimuth correction. User involvement is discussed below.

The early and frequent delivery of builds was the OOS' way of implementing these tenets of the manifesto. Being able to see working code – even if that code initially was focused on architecture and tools rather than interesting military capability – gave the users confidence in the process and the development team. As McCabe and Polen state, "Until a usable system is delivered, the customer has nothing to show for its investment" [3].

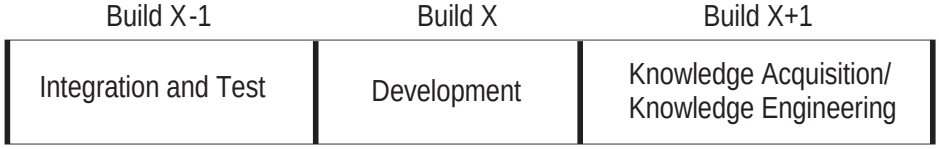
Paulk asserted the following:

> Agile methodologies, with their rapid iterations, require continual planning. Customer collaboration and responsiveness to change are tightly linked, if perhaps inconsistent with typical government-contractor relationships. [7]

The overall, strategic program requirements and overall architecture changed little over the four years of software development; however, the specific, tactical requirements and many of the smaller design decisions were made as late as possible.

The overall goals of a particular block were locked down during the block planning a few weeks before the beginning of the block, not at the beginning of the program in a *big bang*. Additionally, the fine-

Figure 2: *Three-Build Process for Creating a Functional Capability in OOS*



grained requirements and design of a build were finalized only one build before execution. While executing Build *X*, the program is planning Build *X*+1 and testing Build *X*-1. A change in requirements, therefore, could be reacted to within 16 weeks, not a year. Figure 2 illustrates this process.

A major advantage of a spiral methodology was the ability of the program to adapt to requirements changes. McCabe and Polen asserted, "When needs are changing, the value of the original system as specified, however optimum it was at the time of its conception, depreciates daily" [3]. During program development, the Army made a radical change in organization to a focus on Brigade Combat Teams. The OOS program was able to rapidly shift focus between spirals so that it would be delivered with the new force structure rather than the old. Additionally, the threat faced by U.S. forces changed. OOS was able to curtail representation of older, Soviet-style opposing forces and implement a more contemporary, unconventional enemy.

Customer Involvement in Development

Turner and Boehm state:

> One of the major differences between agile and plan-driven methods is that agile methods strongly emphasize having dedicated, collocated customer representatives, while plan-driven methods count on a good deal of up-front, customer-developer work on contractual plans and specifications. [9]

Because the customer representatives are collocated in the IDE, they participate in all our meetings, and are available to answer questions or reach back into their customer base for feedback and input.

Several of the OOS builds were designated User Assessment Baselines (UAB). These UABs were available in the IDE for users to evaluate. In addition, a number of assessment events were held in the IDE during which users from around the Army were invited to participate. Finally, the PM office took the software to user sites for more formal assessments. It may seem obvious to some, but users use the software differently than engineers and developers. While the users often tried to treat these developmental assessments as operational tests, resulting in often vitriolic feedback, the events were excellent opportunities to allow users to identify bugs in the software in an operational-like environment.

Documentation Versus Working Code

OOS software builds can vary between eight and 10 weeks in duration, depending upon the difficulty of the tasks in that build, and how they fit in the timeframe of the block. Since Build 4, OOS has had working software that could be demonstrated and made available for user feedback. As McCabe and Polen state, "Your only real knowledge comes from a working system" [3].

Having working code since Build 4 reflects the agile method's bias toward working code rather than documentation-centric development. However, extensive design documentation, knowledge acquisition documentation, and technical notes do exist and are reposed in <www.OneSAF.net>, the program's collaborative information warehouse. In addition, user documentation is maintained in a form that is simultaneously compiled into a users' manual and online help.

"Being able to see working code – even if that code initially was focused on architecture and tools rather than interesting military capability – gave the users confidence in the process and the development team."

Paulk states that when employing agile methods, a project must:

> Decide where to place the balance point in documentation and planning to alleviate the concerns of the stakeholders (and regulatory requirements) while achieving the flexibility and benefits promised in the agile philosophy. [7]

This is not to imply that the OOS software is not adequately documented. The A&I contractor has been externally certified at Capability Maturity Model Integration (CMMI®) Level 5, so the development processes are well documented. Not only are the processes documented, but the results of the processes (the development products/artifacts) are captured. Metrics are captured on the execution of processes as well, and the A&I contractor conducts periodic *defect prevention* sessions to examine and correct the root cause of common issues.

The documentation of these processes and products is captured in a Web-enabled manual known as the Electronic Process Guide. The artifacts that document developers' adherence to these processes are found in several Web-enabled repositories such as the online Software Development Folders, Web-based tracking tools for action items, trouble reports, defects, peer reviews, and risks. The very nature of having a Web-enabled tool set reduced the burden on developers for complying with these processes and enabled communication across teams.

Extreme Testing

With respect to testing, agile methods indicate the projects should include the following [8]:

- Code the unit test first.
- All code must have unit tests that must pass before being released.
- When a bug is found, tests are created.
- Acceptance tests are run often and the score is published.

Whenever an OOS developer commits his or her changes to configuration management (i.e., Concurrent Versions System), a tool – called BuildBoy – builds the software for Linux, Windows, and Solaris and runs nearly 3,000 automated tests on each operating system. If the build fails, the developer and his or her supervisor are sent an e-mail. The developer can then reference the BuildBoy Web page to determine the nature of the failure and take corrective action. In this way, the software is built and subjected to unit and integration tests on average of eight times a day.

McCabe and Polen state:

> Writing automated unit tests first is a clever way of inducing developers to not only unit test their code efficiently, but also to write their code efficiently without superfluous logic. [3]

While we agree in concept, this is an area in which we need improvement. All code is delivered at the end of a build with appropriate tests, and some of the teams do build the tests first, but we are not consistently using the methodology of building the test first.

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

With respect to the tenet of building a test when a bug is found, we have adapted this process to large, complex software. While we do not build a test for every bug that is fixed, when we find the same pattern of bug appears many times, we build an automated test to trap it.

Conclusion

The OOS software is being developed through a combination of agile, extreme, and traditional techniques. We have not blindly adopted all of the techniques and tenets of these agile approaches; however, we have used a great many of them. The program still has room for improvement in some areas, and our CMMI Level 5 continuous process improvement is ever vigilant for opportunities to do so. In particular, we are in need of more agile testing and an increased number of automated tests.

The unique blend of these techniques has been instrumental in the award-winning success of the OOS development effort. Some readers may wonder which of these techniques were most beneficial and if they could be applied to other programs with equal success. The answer to both of these questions is based upon our original premise: the best method depends upon the nature of the program itself. Program characteristics such as team size and impact of system failure are evident; however, each PM needs to consider other issues that define the technical, programmatic, and political landscape: Will upper management support less conventional techniques? Does the government team have the appropriate skill set to work closely with the software development team, not just monitor contractor activity and check end results? Are the users open to agile methods and willing to actively participate throughout the process?

The best methods are as unique as the programs themselves. Since our program initiation, we have engaged with other organizations that have tried to emulate the form – without the substance – of these agile methods and innovative business processes; their successes have been limited. The authors considered what methods and techniques worked best for the OneSAF program. After pondering this question, the answer illustrates the shades of gray between traditional and agile methods. Strictly following the CMMI Level 5 processes (traditional) and the individual interactions (agile) among users, developers, and government representatives were essential contributors. Four years ago we did not know what would work best, or what would work at all for that matter. However, we were willing to try something innovative, willing to make changes along the way, and bold enough to see it through.◆

References

1. Highsmith, J. "History: The Agile Manifesto." The Agile Alliance, 2001 <http://agilemanifesto.org/history.html>.
2. Agile Alliance. "Agile Software Development Manifesto." Agile Alliance, 13 Feb. 2001 <www.agilemanifesto.org>.
3. McCabe, R., and M. Polen. "Should You Be More Agile?" CROSSTALK Oct. 2002 <www.stsc.hill.af.mil/crosstalk/2002/10/mccabe.html>.
4. Cockburn, Alistair. "Learning From Agile Software Development – Part One." CROSSTALK Oct. 2002 <www.stsc.hill.af.mil/crosstalk/2002/10/cockburn.html>.
5. Parsons, D., and R. Wittman. "Open Source Opens Opportunities for Army's Simulation System." CROSSTALK Jan. 2005 <www.stsc.hill.af.mil/crosstalk/2005/01/0501parsons. html>.
6. Agile Alliance. "Principles Behind the Agile Manifesto." Agile Alliance, 30 May 2005 <http://agilemanifesto.org/principles.html>.
7. Paulk, M. "Agile Methodologies and Process Discipline." CROSSTALK Oct. 2002 <www.stsc.hill.af.mil/crosstalk/2002/10/paulk.html>.
8. Wells, D. "The Rules and Practices of Extreme Programming." Extreme Programming. 28 Feb. 2004 <www.extremeprogramming.org/rules.html>.
9. Turner, R., and B. Boehm. "People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods." CROSSTALK Dec. 2003 <www.stsc.hill.af.mil/crosstalk/2003/12/0312turner.html>.

Did this article pique your interest? You can hear more at the Eighteenth Annual Systems and Software Technology Conference May 1-4, 2006, in Salt Lake City, UT. Doug Parsons will present in Track 7, Room 251A-C on Thursday, May 4, at 8 a.m.

About the Authors

LTC John "Buck" Surdu, Ph.D., is the product manager for the One Semi-Automated Forces Objective System. Originally commissioned as an infantry lieutenant, Surdu served in operational assignments in the 82nd Airborne Division, Europe, and Korea. He worked as a research scientist at the Army Research Laboratory and a senior research scientist and assistant professor in the Information Technology and Operations Center within the Department of Electrical Engineering and Computer Science at West Point. He has a Bachelor of Science in computer science from the United States Military Academy, West Point, a Master of Science in computer science from Florida State University, a Master of Business Administration from Columbus State University, and a doctorate in computer science from Texas A&M University.

12350 Research PKWY
Orlando, FL 32826-3276
Phone: (407) 384-5103
Fax: (321) 235-1484
E-mail: john.surdu@us.army.mil

Doug J. Parsons is the lead engineer of the Intelligent Simulation Systems Team at the U.S. Army Program Executive Office for Simulation, Training, and Instrumentation. His primary focus is toward the successful development of the One Semi-Automated Forces Objective System. Parsons has a Bachelor of Science in mechanical engineering from North Dakota State University, a Master of Science in systems management from Florida Institute of Technology, and a Master of Science in industrial engineering from the University of Central Florida.

Program Executive Office – Simulation Training and Instrumentation (PEO-STRI)
12350 Research PKWY
Orlando, FL 32826-3276
Phone: (407) 384-3821
E-mail: doug.parsons@us.army.mil

Managing Cultural Changes in Your Organization

Dr. Kenneth D. Shere
The Aerospace Corporation

Maturity models are great; they provide a mostly technical road map for what we need to do to improve processes. Lean Six Sigma is also great – it provides a methodology for how to improve processes. The problem is that process improvement causes cultural changes. This article provides guidelines for managing cultural changes in your organization.

In earlier articles [1, 2], discussions on how Lean Six Sigma (LSS) affects the government, and how LSS compares to the Capability Maturity Model® (CMM®) have been presented. Key aspects of why LSS has been successful include the following:

1. It has an external focus based on listening to the voice of the customer.
2. The cost of poor quality is explicitly considered.
3. LSS is essentially good systems engineering in which the methodology is institutionalized.
4. Training is an inherent part of the methodology.

Many articles on capability maturity models have been published in CROSSTALK. The internal focus and structure of these are complementary to LSS. Both approaches have been used successfully, and when combined provide a powerful basis for producing effective products with low time and budget variances.

This claim has been recognized. For specific examples, the reader can search the Software Engineering Institute Web site at <www.sei.cmu.edu>. One of the companies publishing results on that site is Raytheon. Specifically, it is their North Texas Software division, located in Plano and McKinney.

While these methods are useful and there exists substantial data from both government oriented programs and purely commercial programs to prove it, they inherently involve changing the way organizations conduct business.

The change process is very taxing. People at all levels of an organization tend to resist change. This article explores why change is necessary, and provides some information on what it takes to enact change. This article is intended for leaders of organizations who are grappling with change. The topics discussed are the following: change is hard, facilitating change, strategic thinking, and first-year strategy. There are many books and journal articles written on these topics. A high-level discussion is provided here to provoke thought and additional reading.

Change Is Hard

Change affects people's lives. Reaction to change varies from the view expressed by the comic strip character, Pogo, "The certainty of misery is better than the misery of uncertainty," to the view of the late Jerry Garcia of the rock band The Grateful Dead, "Somebody has to do something, and it's incredibly pathetic that it has to be us" [3]. To successfully implement change, it is necessary to understand the urgency for change and to adopt a positive attitude about change. Understanding how change affects people in an organization is a key component of breaking down resistance.

"To successfully implement change, it is necessary to understand the urgency for change, and to adopt a positive attitude about change. Understanding how change affects the people in an organization is a key component of breaking down resistance."

Some of the causes for the urgency of change could be the following:

- Your customers are dissatisfied (and may be considering alternative sources).
- Budget issues.
- Commission reports such as the Space Commission [4], or reports of the Defense Science Board.
- Validity of your organization's business assumptions.
- Self-inflicted pain.

Many organizations do not have a clear understanding of who are their customers, partners, and stakeholders. Consequently, it is difficult to know underlying causes for dissatisfaction and key issues that need to be addressed. In this article, the term *customers* refers to stakeholders, partners, service or product users, and operators.

Dissatisfaction occurs at many levels and for a variety of reasons. Sometimes an organization thinks that everything is fine because its average product or service quality is high, but people do not experience the average – they experience the variability. For example, telephone service is expected to be excellent. If a builder accidentally cuts a telephone cable, leaving thousands of users without service for five days, the telephone company is going to have a lot of unhappy customers. Customers might understand a one-day outage, but rapidly lose patience with longer periods. In areas where alternative service is provided by cable television companies, the telephone company can lose substantial business through no fault of its own.

Determining customer satisfaction as a predictive measure is very difficult. Organizations need to conduct surveys, talk to customers, and collect data. A company's sales are a backward-looking indicator of customer satisfaction. When people stop buying the product, it is frequently too late to regain market share. The current state of the American auto industry is a classic example of the result of customer dissatisfaction.

Sometimes customers are dissatisfied because they do not fully understand the services provided. Other times, dissatisfaction might be related to budget issues. Each organization needs to approach customer satisfaction from a variety of perspectives.

Typical budget issues include not understanding the cost of doing business, and being unable to defend an organization's budget to Congress. The latter issue

is related to stakeholders not understanding all the services provided and the value of these services.

Being able to demonstrate that an organization's processes are lean can go a long way toward defending its budget. It enables the organization to specify the services that will be lost as a function of budget reduction.

This concept also applies to acquisition. For example, if an organization were to acquire the next-generation weather satellite and experienced a significant budget cut, the organization should be able to tell Congress of the impact. The organization should be able to specify how this reduction affects the products that can be produced, and how these products are related to weather phenomena, agribusiness, transportation, and energy.

Reasons for budget reductions include cost overruns by major programs (in other parts of the agency or a related agency) and pressures to balance a budget. The latter is especially prevalent in years with major national catastrophes such as Hurricane Katrina and years in which wars are being fought.

Sometimes the urgency for change is provided by independent commission reports. For example, the Space Commission reported on significant problems associated with space programs and made a number of suggestions related to U.S. Air Force space programs and the National Reconnaissance Office (NRO). This report stated that the NRO has lost its edge. The chairman of this committee, Donald Rumsfeld, needed to resign a little more than one week before the report was delivered to the secretary of defense because he was to be sworn in as the secretary of defense. Needless to say, changes have occurred as a result of this report.

Space Commission reports and other congressional actions are indicators that the organization's business environment has changed – that its business assumptions may no longer be valid. There are many examples of organizations – sometimes whole industries – that fail to understand the changes to their environment. The result is frequently obsolete or incorrect business models. Examples include the following:

- The failure of General Motors (GM) to recognize competition from Japan (and other issues).
- The bankruptcy or near bankruptcy of all major airlines because they use a hub-and-spoke system and have very high wages. These airlines simply cannot compete with the business model of discount airlines represented by Jet Blue and Southwest.
- Full-service stock brokerage firms have lost substantial business to discount firms and online trading.
- America Online (AOL) is still trying to understand what its business should be in the light of high-speed modems and excellent, free search engines. Google recently purchased 10 percent of AOL's stock and could become the driver for defining the AOL business model.
- Wang computers had a virtual stranglehold on word processing and office automation in the 1970s. They insisted on maintaining a proprietary system and could not survive the competition of Microsoft Word and Word Perfect.

The final cause for urgency of change discussed here is self-inflicted pain, which comes from a variety of sources. A primary source of self-inflicted pain is making it difficult to conduct business with your organization. If your organization provides services, how does it compare with the expected service that people receive from national companies like Nieman Marcus or Marriott? Even though the business areas are different, your organization can be held to that standard because your customers and stakeholders might patronize these companies.

"Without fully committed leaders, any process improvement or change initiative will fail. It is simply not worth proceeding if you do not have this type of commitment."

Sometimes, self-inflicted pain arises from customers and stakeholders not understanding the services provided. These situations might be resolved through a good education or marketing effort. Other sources of self-inflicted pain are derived from leadership actions. For example, in one report a Navy captain ran a guided missile cruiser over a large submerged rock, injuring several crew members. Sailors on the bridge suspected the ship was headed for the rock, but they were afraid to tell the captain to change course for fear of being wrong [4].

As you look at the urgency for change, tie these various causes together. Would the government be better off outsourcing your services? The answer is usually no, but it needs to be supported with data.

Facilitating Change

In a related article, the co-author and I stated, "The three most important approaches to changing culture are communications, communications, communications" [5]. *My new and improved list* consists of 12 items, but the first nine are heavily intertwined:

1. Leadership.
2. Leadership.
3. Leadership.
4. Commitment.
5. Commitment.
6. Commitment.
7. Communications.
8. Communications.
9. Communications
10. Strategic thinking.
11. Consistency.
12. Understanding the data.

Without fully committed leaders, any process improvement or change initiative will fail. It is simply not worth proceeding if you do not have this type of commitment. The leader of a small group might be able to improve the processes used by his or her group, but a sustained major change requires commitment at the top.

One day in 1993, I bumped into an acquaintance who was president and chief operating officer of a major corporation. He said that he was leaving the next day to attend a weeklong conference on business process reengineering (BPR). Think about the cost of attending that conference. That week corresponds to 2 percent of his time for the entire year. Think about all of the other items commanding his attention that will not be done during that week. Surely every vice president of that company knew that he was attending the conference, and understood how seriously he was taking BPR.

Somewhat over a year later, I asked this person about his accomplishments. The results were somewhat amazing – he completely reengineered a major production process (investing about $125 million) and entered two new, but related, businesses. Success was achieved because of leadership and commitment at the top.

Communication is also critical to success. To enact change, it is necessary to talk to *everyone* in the organization and convey the urgency for change. It is also necessary to listen to people and to understand their concerns. These concerns can

be somewhat relieved by telling a story about the future. Describe your vision and why it is necessary. The Declaration of Independence tells the story of why change was necessary. The U.S. Constitution is a formal story of the political utopia envisioned by our founding fathers [6]. These stories tell the urgency for change and how people will be affected by the change.

Resistance is normal. Recipients of bad news go through a series of psychological states: denial, anger, bargaining, despair, and acceptance. In the final state they can act. Change leaders need to understand resistance to change and to communicate with people in ways that mitigate their resistance. Much of this communication involves explaining to people how they will be affected by the change.

Gen. Gordon Sullivan (retired) discusses leadership resistance in terms of three traps: doing things too well, being in the wrong business, and making yesterday perfect [7].

When a company is being run well, it is hard to recognize the need for change. Before Jack Welch became chief executive officer (CEO) of General Electric (GE), his predecessor was viewed as the top CEO in the country. Every management school viewed GE as an example because they were highly diversified, immune to the ups and downs of business cycles, and regularly grew between 2 percent and 3 percent per year. It was as good as it gets.

Welch recognized the need for change. He immediately said that GE performance was horrid. GE needed to grow at double-digit rates. GE went through a dramatic change, going from 80 percent manufacturing and 20 percent services in 1982 to 80 percent services and 20 percent manufacturing in 2002. During this 20-year period, the company's stock price soared.

Being in the wrong business consists of not understanding the implications of change. Alfred Sloan said that GM was in the business of making money, not cars. He also said that success comes not from technological leadership, but from having the resources to quickly adopt the innovations successfully introduced by others [8]. Consider the situation of GM today – their bond rating has been reduced to junk.

Making yesterday perfect causes people to ignore or rationalize data about their environment. In military terms, making yesterday perfect corresponds to fighting yesterday's war. The entire thrust of moving the U.S. Army into the digital age – the item that occupied most of Sullivan's time when he was chief of staff – has been to avoid future failure by being prepared for future wars.

Summarizing this section, process improvement involves change. Leaders must be committed to change and be able to communicate the urgency for change to everybody in their organization. They need to listen, think strategically, have a reward system that is consistent with their goals, and understand the environmental data to which they are exposed.

Strategic Thinking

Strategic thinking, which in a sense encompasses leadership, commitment, and communications, is best described in terms of the trigraph depicted in Figure 1[1]. In terms of change, the order of strategic thinking is strategy, structure, and culture. Culture and structure are briefly discussed. Most of the discussion in this section focuses on strategy because that comes first.

There are a variety of definitions for culture, but most of them are related to growth or transmitted behavior. At the workplace, culture is embodied in the rules taught to the next generation of workers. Understanding changes to these rules for success takes a long time. People need to observe whether the rewards are actually in accordance with the communicated changes. They also need to observe changes in the organization's structure.

It is reasonable to expect culture change to take from five to seven years to be fully realized. Culture change depends on changes to the organizational structure and to the strategy. The amount of time it takes for change cited here is based on the author's observations and general reading. The author is unaware of specific studies that identify time required for change.

The most important step for changing culture is to walk the talk (pardon the pun). Employees always know the truth because they observe leaders' actions. In 1989, a survey showed that 43 percent of employees believed that management lies and cheats. Another survey taken in 1992 showed that 64 percent of the employees think that management lies [9].

The organization's actual values are communicated through actions – not posters. For example, does an organization reward firefighters or fire marshals? If firefighters are rewarded, do not expect the employees to buy into process improvement that stresses getting the job done right the first time.

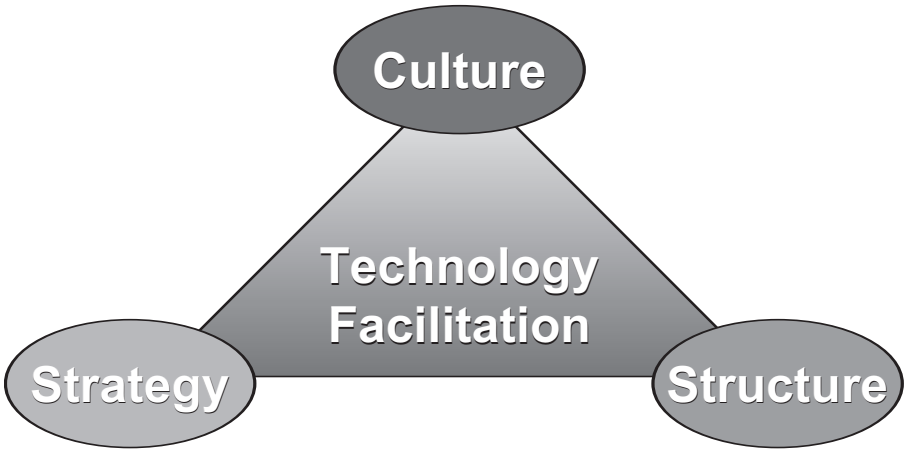
The structure of an organization consists of organizational charts (i.e., reporting), roles, and relationships. During the first year of change, an organization generally focuses on strategy – values, goals, and vision. Aligning an organization with its strategy generally takes one to two additional years. Leaders need to think about where they want to be in 10 years. They then need to determine the organizational structure that is needed to achieve their goals and vision.

The structure of the organization usually needs to be modified so that the right people are doing the right jobs. Sometimes, this is a matter of reassigning the right person who has been in the wrong job. Other times it is necessary to bring in new people, either through promotion or from outside the organization. This is the tough part of changing the structure.

Using GE as an example of a company that has undergone major cultural changes over the years, it is interesting to observe that GE has never brought in a new CEO from the outside. Jack Welch was a lifelong GE employee. His replacement, Jeff Immelt, was promoted from within.

Change can occur from within an organization by having the right people in the right positions. Jack Welch used LSS as a tool to help him create cultural change. At GE's 1999 annual meeting, he said that corporate strategies for the foreseeable future were globalization, services, and Six Sigma.

Figure 1: *The Strategic Thinking Trigraph*



If Six Sigma or CMM Integration[SM] is part of the strategy, how is this to be done from an organizational perspective? Is it part of a corporate quality group? Does each division need to have a process czar? Even when we know what needs to be done, it is difficult to enact these changes in industry, and much more difficult in government. It frequently requires leaders to act in the best interest of the person who will come after them. For example, VADM Phillip Balisle decided to implement LSS throughout the Naval Sea Systems Command. Not long after Balisle made this decision, VADM Paul Sullivan replaced him. Sullivan has also committed to LSS and is implementing it throughout his organization.

First-Year Strategy

The strategy of the organization is generally defined in terms of its mission, core values, vision, and goals. Organizational leaders are generally pretty good at thinking in these terms; however, getting the entire senior leadership team to have a common understanding generally takes about a year.

Holding a series of off-site meetings with advance reading and homework is a primary process for achieving this understanding. These off-site meetings need to be reinforced through time dedicated to strategic thinking at regular staff meetings. The first off-site meeting should focus on establishing a common understanding of the organization's mission and values. The mission is usually understood in government. Values are usually only implicitly known. Each leader thinks that he or she knows them, and in most good organizations there is considerable overlap in what people think. However, this off-site meeting is likely to reveal significant differences. These differences form the basis for extensive discussion of the organization's true values.

A couple of good reading assignments to be completed before the start of this off-site meeting are chapters three through five (on leadership and values) of Sullivan and Harper [10] and Larkin and Larkin [11].

Other topics to be covered in this first off-site meeting include the urgency for change and a brutally honest discussion of the organization's strengths and weaknesses. For example, discuss the reasons for recent successes and failures. Perform a strengths, weaknesses, opportunities, and threats analysis.

Discuss the reading assignments. Have everybody do an exercise to specify why they agree or disagree with Gen. Sullivan's rule No. 2, "Leadership begins with values." Include discussion of leadership styles as part of this session. How do these styles reflect the organization's values? Other questions to ask include:

- How do these values speak to the future of the organization?
- How does organizational and individual behavior reflect these values?
- What behaviors are contradictory to these values?

Subsequent sessions can focus on vision. A leader's vision is a view of what his or her organization and products (or services, etc.) will look like in 10 years. It is critical that the vision and core values be in harmony. Each member of the senior leadership team should write a story that describes the vision in terms of his or her business area or function. Telling stories is a great way to learn [12].

"Leaders must be committed to the change and be able to communicate the urgency for change to everybody in their organization."

Core values and vision act like fields – electric, magnetic, gravitational, etc. They are unseen, but are real forces reflected throughout the organization. (See [13] for further discussion of this and related concepts.)

Consider the Tylenol scare of 1982 and Johnson & Johnson's reaction. Seven people died on Chicago's west side when they ingested extra-strength Tylenol capsules laced with cyanide. Even though this problem was not caused by Johnson & Johnson, they instantly pulled the product from the market and kept it off the market until tamper-proof packaging (and caplets) was developed. It cost the company more than $100 million.

When asked how the company was able to respond so quickly, the CEO of Johnson & Johnson said that not doing so would have violated its core values. He did not need to call meetings; he just applied their core values [14].

The CEO's actions were highly visible to both employees and the public. Every employee in this global company knew what happened. Within three months, Tylenol regained 95 percent of its market share.

Communicate your values and vision to everyone in your organization.

The senior leadership team also needs to define goals, activities, and metrics. Goals are long-term and when achieved, an organization's vision will be substantially realized. Activities are near-term tasks such as plans for the next fiscal year and are tied to the goals. Metrics measure progress toward achieving the goals.

LSS is an example of an activity. Depending on the goals, it could be aimed toward making the organization world-class, reducing costs to meet future budgets, or providing value to customers. Whatever its aim, justify it by measuring results.

Summary

This article focuses on the leadership actions needed to implement process improvement, which involves changing an organization's culture. That topic is generally not discussed when we espouse CMM, LSS, or any other process improvement method yet it is critical to successful implementation. This attempted culture change will cause the ultimate demise of process improvement efforts unless it is handled well.

For those reasons, leadership is the focus of this article. It is pointed out that change is hard. Successful change requires committed leaders who communicate with their employees and have action consistent with what they say. Leaders need to do a lot of strategic thinking. They need to recognize that change occurs over time. Patience is necessary. Leaders need to establish their strategy during the first year and develop an appropriate organizational structure within three years. Then, through consistency, paying attention to the data, and communication, the culture will change.

This approach is encapsulated in the title of Peter Schwartz's book "The Art of the Long View."◆

References

1. Shere, Kenneth. "Lean Six Sigma: How Does It Affect the Government?" CrossTalk Mar. 2003: 8-11 <www.stsc.hill.af.mil/crosstalk/2003/03/shere.html>.
2. Shere, Kenneth. "Comparing Lean Six Sigma to the Capability Maturity Model." CrossTalk Sept. 2003: 9-12 <www.stsc.hill.af.mil/crosstalk/2003/09/0309shere.html>.
3. Pritchett, Price. The Employee Handbook of New Work Habits for a Radically Changing World. Pritchett

[SM] CMM Integration is a registered service mark of Carnegie Mellon University.

Rummler-Brache, 1999.
4. National Security Association. Report of the Commission to Assess United States National Security Space Management and Organization. Washington, D.C.: NSA, 11 Jan. 2001.
5. Shere, Kenneth D., and Scott R. Turner. "Vision and Cultural Change in Government Satellite Programs." Program Manager. May-June 2003: 88-92.
6. Schwartz, Peter. The Art of the Long View. Doubleday, 1991.
7. Sullivan, Gordon, and Michael Harper. Hope Is Not a Method. New York: Broadway Books, 1997.
8. O'Toole, James. Leading Change. San Francisco, CA: Jossey-Bass Publishers, 1995.
9. Larkin, T.J., and Sandar Larkin. "Reaching and Changing Front Line Employees." Harvard Business Review. May 1996.
10. Sullivan.
11. Larkin: 95-104.
12. Kaye, Beverly, and Betsy Jacobson. "True Tales and Tall Tales: The Power of Organizational Storytelling." Training and Development. Mar. 1999: 45-50.
13. Wheatley, Margaret. Leadership and the New Science. Berrett-Koehler Publishers, 1999.
14. Tichy, Noel M. The Leadership Engine. New York: Harper Business, 1997: 116.

Note

1. The original source of this figure is uncertain, but I was first shown it by Sheila Sheinberg, a nationally known consultant in organizational behavior. Some of the ideas in this article come from working with her.

About the Author

Kenneth D. Shere, Ph.D., is a senior engineering specialist at The Aerospace Corporation where he provides systems and software engineering, acquisition, and strategic leadership support to various governmental organizations. He is certified as a Lean Six Sigma green belt and a Software Engineering Institute Software Capability Evaluator. He has a Bachelor of Science in aeronautical and astronautical engineering, a Master of Science in mathematics, and a doctorate in applied mathematics, all from the University of Illinois.

The Aerospace Corporation
1000 Wilson BLVD STE 2600
Arlington, VA 22209
Phone: (703) 608-0904
Fax: (703) 812-9415
E-mail: shere@aero.org

CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:

Software Assurance
September 2006
Submission Deadline: April 17

Star Wars to Star Trek: How Science Fiction Affects Real World Technology
October 2006
Submission Deadline: May 22

Back to Basics/ Management Basics
November 2006
Submission Deadline: June 19

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <www.stsc.hill.af.mil/crosstalk>. We accept article submissions on all software-related topics at any time, along with Letters to the Editor and BackTalk.

COMING EVENTS

May 1-4
2006 Systems and Software Technology Conference
Salt Lake City, UT
www.stc-online.org

May 1-5
PSQT 2006 West
Practical Software Quality and Testing
Las Vegas, NV
www.psqtconference.com/2006west

May 7-11
2006 North America Computer Audit, Control and Security Conference
Orlando, FL
www.isaca.org

May 9-10
Fulfilling the Warfighter's Vision 2006 Closing the Information Gap
St. Petersburg, FL
www.afei.org/brochure/6a06/index.cfm

May 16-17
MEECC 2006
Military Embedded Electronics and Computing Conference
Long Beach, CA
www.meecc.com

May 20-28
28th International Conference on Software Engineering
Shanghai, China
www.isr.uci.edu/icse-06

June 19-22
CISC 2006 Combat Identification Systems Conference
Orlando, FL
www.usasymposium.com/cisc

June 25-30
18th Annual Forum of Incident Response and Security Teams Conference on Computer Security and Incident Handling
Baltimore, MD
www.first.org/conference/2006

Cooperative Appraisals for Capability and Risk Evaluation

Diane A. Glaser
U.S. Army

Michael D. Barnett
MTC Technologies

The U.S. Army Communications-Electronics Life Cycle Management Command Software Engineering Center is working with the Software Engineering Institute in creating a framework for cooperative government/industry appraisals for process improvement and risk evaluation. Traditionally, solicitations for Department of Defense projects have included some sort of risk evaluation. Though risk evaluations are only one component of a source selection, all bidders underwent the risk evaluation site visit, costing the government significant time and effort in evaluating potentially several organizations that would not perform the work solicited. The concept developed is to partially base an award on the merits of a process proposal with the understanding that an on-site evaluation would follow after contract award. Another aspect of these appraisals is that representatives from the government and the organization being appraised work together on the appraisal team to jointly evaluate the organization. This article describes the efforts of the authors in performing post-award cooperative appraisals, the lessons learned, and the benefits to both the government and the appraised organization.

Traditionally, solicitations for Department of Defense (DoD) projects have included some method of risk evaluation to determine the level of risk that the project manager (PM) will face in selecting a bidder to provide the products/services for his or her program. This risk evaluation could take the form of a Software Capability Evaluation (SCE[SM]), a methodology developed by the Software Engineering Institute (SEI[SM]) at Carnegie Mellon University, or other methods.

The Communications-Electronics Life Cycle Management Command (C-E LCMC) Software Engineering Center (SEC) developed a streamlined form of the evaluation called the Software Process Risk Evaluation (SPRE) that has been mandated for all major C-E LCMC acquisitions. This method, like the SCE, has been used during the solicitation process to evaluate all potential vendors and provide input to the evaluation factors for the solicitation.

There were several problems with the SCE and SPRE methods. There often was not enough time to prepare for the evaluation. A lack of historical information about the bidders' processes made the evaluations more critical while decreasing the effectiveness of a short, intense on-site visit. The cost of the evaluations were high for both the government, who has to visit all of the bidders during the proposal evaluation period, and the contractor, who has to apply significant resources to prepare data and provide people to be interviewed during the on-site visit. Both parties potentially spend money to support multiple source selection evaluations. The government expends resources to evaluate the losing contractors, and the contractor could possibly have to support multiple evaluations in a given timeframe.

[SM] CMM Integration, SCE, SCAMPI, and SEI are service marks of Carnegie Mellon University.

Acquisition Reform

Industry has matured over the years. The SEI issued the Capability Maturity Model® for Software (SW-CMM®) in 1993. Since that time, many of the organizations that bid on acquisition solicitations have undergone process improvement initiatives using SW-CMM or its successors, including the CMM Integration[SM] (CMMI®). With the advent of the CMMI, a new appraisal method – the Standard CMMI Appraisal Method for Process Improvement (SCAMPI[SM]) [1] – has been developed and is quickly becoming the appraisal method of choice, regardless of which CMM is being used as the model for process improvement. SCAMPI can be used in lieu of the SCE/SPRE for evaluating bidders during a source selection.

Government acquisition reform has evolved as well. The Interim Defense Acquisition Guidebook, paragraph C5.2.3.5.6.1.5, states the following:

> Select contractors with domain experience in developing comparable software systems; with successful past performance; and with a mature software development capability and process. Contractors performing software development or upgrade(s) for use in an [Acquisition Category] ACAT I or ACAT IA program shall undergo an evaluation, using either the tools developed by the [SEI] or those approved by both the DoD Components and the Deputy Director, Software Intensive Systems. At a minimum, full compliance with SEI Capability Maturity Model Level 3, or its equivalent in an approved evaluation tool, is the department's goal. However, if the prospective contractor does not meet full compliance, risk mitigation planning shall describe, in detail, the schedule and actions that will be taken to remove deficiencies uncovered in the evaluation process. Risk mitigation planning shall require Product Manager approval. The Deputy Director, Software Intensive Systems shall define Level 3 equivalence for approved evaluation tools. The evaluation shall examine the business unit proposed to perform the work. The reuse of existing evaluation results performed within a 2-year period prior to the date of the government solicitation is encouraged. [2]

Later guidance clarified Level 3 equivalence. The Software Development Capability Evaluation [3], developed by the U.S. Air Force with approved core set revisions, was determined to be an acceptable alternative set of criteria to the SW-CMM. The use of CMMI Systems Engineering and Software Engineering Vers. 1.1 Level 3 criteria is another acceptable alternative to the use of SW-CMM Level 3 criteria.

Newer guidance from the Office of the Secretary of Defense/Acquisition, Technology, and Logistics no longer requires Level 3 equivalent ratings. Process improvement is encouraged, but maturity levels are no longer stated in the guidance. The rationale is that many organizations have reached maturity Level 3 or better, and that the original

direction was not to have organizations go for the rating, but to initiate organizational process improvement.

The CMMI, in its continuous representation, allows an organization to be appraised to determine a process capability profile; the organization can look at Process Areas (PAs) of interest based on organizational goals and see how well they have implemented them. This can also allow a PM to focus on areas that are of high risk for a particular acquisition, rather than look at overall maturity across all PAs.

Appraisal Reuse

To reduce costs and resources associated with evaluating development capabilities, the DoD is working with industry to provide implementing mechanisms that better support the reuse of appraisals. Results of previous appraisals conducted on the organizational unit proposing to do the work may be an acceptable alternative to the government performing a new appraisal. The reused appraisals must be shown to be independent, i.e., at a minimum, the lead appraiser should not be from within the organization being appraised.

C-E LCMC Acquisition Strategy

The SEC has been involved in the examination of cooperative government/industry appraisals as an alternative to the conventional acquisition strategy for several years [4]. The SEC took the initiative in working with the acquisition center at Fort Monmouth to identify software-intensive acquisitions occurring there. The interim project office was contacted and cost-effective strategies developed to address the acquisition agent's need to reduce risk. One of these strategies was to wait until after contract award and perform a post-award appraisal of the successful bidder.

To make the post-award appraisal viable, the solicitation package must have the appropriate language. The contract should require that the product be developed using CMM/CMMI Level 3 processes. Proof is to be submitted, demonstrating that the contractor is rated as a Level 3 organization (e.g., copies of CMM/CMMI appraisals and process improvement track record). If the organization cannot verify Level 3, a detailed process improvement plan, including a schedule that leads to a Level 3 appraisal is submitted. This provides the acquisition agent with material in which to evaluate the bidders' process maturity during the source selection without the effort required for on-site visits.

The post-award appraisal must also be contained in the contract. Typically, C-E LCMC requires that the contractor is responsible for leading and conducting an appraisal with government participation on the appraisal team. This is appropriate as it is the contractor's plant and processes. The contractor is required to submit an appraisal plan or process proposal as a formal deliverable after the contract is awarded.

If a CMMI SCAMPI appraisal is selected or required, additional options present themselves. The SCAMPI framework defines three classes of appraisal. A Class C appraisal is a very cursory look at the processes of the organization. It can often be nothing more than a review of process documentation and its application to the project. A Class B appraisal is more robust, but does not emphasize the depth of coverage and rigor that result in a maturity level rating for the organization. Class A appraisals are performed by SEI-authorized SCAMPI lead appraisers. They can lead to a formal maturity rating or capability profile that is submitted to the SEI. The government determines which class SCAMPI appraisal is appropriate for the acquisition.

"The CMMI, in its continuous representation, allows an organization to be appraised to determine a process capability profile; the organization can look at Process Areas (PAs) of interest based on organizational goals and see how well they have implemented them."

With the SEI developing formal mechanisms for SCAMPI B and C appraisals, this methodology is fitting into a DoD approach suggested by Mark Schaeffer, the DoD sponsor for CMMI, and Director, Systems Engineering for the Office of the Secretary of Defense. This approach, reported by Mike Phillips in "CMMI: A Progress Report" [5], would have acquirers look for strengths or weaknesses in the development processes that constitute risks to the proposed development effort. Satisfying CMMI goals, process areas, or maturity or capability levels would not be the point; it is primarily a risk identification and mitigation approach.

Schaeffer suggests a three-phased approach. In phase 1, bidders would be appraised for PAs that the Government Program Office considers highest risk. After contract award, in phase 2, the winning team undergoes a baseline appraisal, using a risk-based analysis of the process strengths and weaknesses, thereby establishing action plans for future checks. In a risk-based appraisal framework, the CMMI is used to identify and group weaknesses to address systemic problems. The CMMI's process categories – Project Management, Engineering, Support, and Process Management – may be used to define areas that the PM believes most affects the developer team's contract performance. Using these categories effectively covers both product and process risk. In phase 3, the risks are monitored to closure.

Tying CMMI/risk mitigation to an award fee on a contract vehicle can be a good incentive for the contractor. A program office should consider wording the contract to have a continuous process monitoring function. For example, a PM could elect to utilize the Defense Contract Management Agency (DCMA) or some other matrix organization to perform process monitoring. This is also a recommendation made in the Workshop on CMMI Use in Acquisition [6].

Benefits of Post-Award Appraisals

The PM saves resources by limiting the number of appraisals required for a solicitation. Maturity risk is only one of many evaluation factors for an acquisition source selection evaluation. The increased risk of not performing an on-site evaluation for each bid is mitigated by the cost and time savings. Many organizations have been performing process improvement initiatives for more than 10 years. This legacy of process improvement tends to ensure that processes are defined, which is the focus of maturity Level 3 in the CMM/CMMI.

The bidders benefit in this approach as well. Organizations can reuse their

Attribute	SCE/SPRE/SCAMPI	Cooperative Appraisal
Timeframe	▪ Pre-award. ▪ May be used post-award for contract monitoring.	▪ Pre-award or post-award. ▪ Recommend post-award baseline and follow-on for contract monitoring. ▪ Can be linked to supplier's process improvement appraisal schedule.
Cost	▪ High (both supplier and acquirer).	▪ Lower (cost sharing).
Cost Effectiveness	▪ Low, especially if supplier must support evaluations for losing solicitation.	▪ High. ▪ Eliminates cost of evaluating losing bidders.
Resources Needed	▪ Resource-intensive.	▪ Resource sharing for staffing appraisals. ▪ Supplier can reuse process data.
Incentive for Process Improvement	▪ Little if no contractual language for process improvement.	▪ Supplier can reuse process assets developed for process improvement. ▪ Potential for appraisal reuse on subsequent solicitations.
Integrated Product Development	▪ Government-only team does not form integrated teams with supplier.	▪ Fosters early development of integrated teams. ▪ Facilitates government/supplier communication.
Risk	▪ Mitigates selection of high-risk bidders. ▪ Does not allow for an in depth risk evaluation of the supplier unless applied post-award.	▪ Early risk mitigation if performed early in contract execution. ▪ Continuous risk monitoring and control if used for contract monitoring.

Table 1: *Evaluation and Cooperative Appraisal Attributes*

process data for multiple solicitations. They do not have the expense of preparing for an on-site visit for a solicitation that they are not guaranteed to win. Cooperative appraisals reward the organization for its process improvement efforts, thereby encouraging internal process improvement activities. Table 1 summarizes the attributes and how conventional evaluations compare/contrast to the cooperative appraisal methodology.

Pilot Effort for Cooperative Appraisals

The SEC was able to participate in an ACAT I acquisition where the PM was consulted before the release of the Statement of Work (SOW). After being briefed on the benefits of performing a post-award appraisal, the PM agreed to have language added to the SOW for a CMMI self-assessment of all major contributors to the product development effort using the SCAMPI method with up to four government participants.

The winning contractor team submitted an assessment plan to have four teammates undergo a self-assessment. The original plan called out in the SOW was to submit the plan within 60 days after contract award and perform the self-assessments within 120 days after contract award. The assessment would establish a process capability baseline on the organizational unit doing the work on the program as well as determine risk areas to be monitored over the acquisition life cycle. Since the contract was awarded to a team of contractors, consideration was given to the timing requirements for both the plan and the conduct of the self-assessments. The SOW provided a list of CMMI PAs of interest. These were the minimum set of processes that were to be assessed, which tied into the program office's key performance parameters, goals, and objectives for the program.

Using multi-organizational teams brings a new risk to the conventional acquisition evaluation methodology. The contracting team may have individually performed process improvement efforts, but the team does not necessarily share institutionalized processes. Former assessments are not directly applicable to the newly formed team and are less predictive of the maturity or capability of the team doing the work.

In this pilot, the prime contractor worked to develop capstone processes that all teammates would follow, but additionally allowed for the individual organization's processes to be used. For example, a project Software Development Plan (SDP) was written by the prime, which called out use of the other organizations' SDPs for their software contributions to the project.

The contractor's self-assessment plan called for self-assessments to be held at multiple organizations. Each self-assessment covered all maturity Level 2 and 3 PAs. Some sites added higher maturity PAs. The government worked with the contractor team to try to leverage each organization's internal process improvement activities. All of the contractors were transitioning from SW-CMM to CMMI; they were already implementing organizational process improvement plans.

Where feasible, the government allowed the organization to add the acquisition project to the list of internal projects being appraised to avoid unnecessary effort of preparing for an independent assessment of one program. For at least one organization, this meant scheduling the appraisal as part of their externally led SCAMPI. This benefited the government as well, since the project was not yet fully under way and, therefore, many PAs (e.g., technical solution, verification, and validation) could not be rigorously assessed – the project simply was not to that point in its life cycle. The addition of projects in different stages of maturity provided a more rounded picture of the organization and how its institutionalized processes would be applied to a future project.

The SEC provided the two authors as the core team that participated on the self-assessments. Using available local DCMA representatives augmented government participation. The PM had contracted with DCMA to perform process monitoring over the course of the development effort, so their participation in the self-assessments acted as a kick-start in understanding the details of the project's processes.

Cooperative Appraisals and the CMMI

The way that acquisition reform is evolving is similar to that of the evolution from SW-CMM to CMMI. CMMI provides integration of software with other disciplines, i.e., systems engineering and integrated product and process development (IPPD), manifested in integrated product teams (IPTs).

There is an analogous evolution in acquisition – a paradigm shift from the traditional buyer-seller relationship to that of an IPT where both the government and contractor share responsibility for the end product. This integrated team concept necessitates a team approach to assessment as well. In a cooperative appraisal, the government participants add objectivity and diverse experience. The team that is formed during cooperative appraisals can be extended to established IPTs for the acquisition life cycle, where

government participants in the assessment can also be members of the government's project management team.

Anyone who has participated in an SEI appraisal can attest to the fact that the intensity of the shared experience does much to foster team building. A foundation of mutual respect, equality, and cooperation is established throughout an appraisal. These sentiments can be brought back to the project management office and used to continuously facilitate communication between the government and contractor. The cooperative appraisal process allows both parties to utilize and benefit from the appraisal data; there is open process communication between the government and their contractor. This gives the government an understanding of the way the contractor does business.

The government gets to meet the key players, from the president of the corporation to the practitioners at the developer's site, and may additionally see the facilities and operations for the project. This interaction establishes positive working relationships and provides a greater understanding of what the government/contractor IPT brings to the program.

In a cooperative appraisal, the government leverages the project and process expertise brought by the contractor, thus facilitating the government's assessment efforts. Usually there are appraisal team members that are part of the contractor's process improvement group or belong to the team working on one of the projects being appraised. These people can quickly guide the government appraisal team member to the appropriate process artifacts or answer appraisal-focused questions.

Lessons Learned

There was some trepidation on the part of the contractors in performing the cooperative appraisal. It was initially perceived that the government representatives were coming in to perform an audit or evaluation, not to participate in an internal process appraisal. Discussions with the site coordinator and contractor management before the appraisal helped overcome these perceptions.

The government representatives explained that the cooperative appraisals were one more manifestation of the IPT method of running a program where the government and the contractor share responsibility for the program's success. The contractors were reminded that their organization's formal appraisal teams are often composed of personnel external to the organization, and that the government representatives' goal was to be integral members of the team. The measure of success would be how forthcoming those interviewed were during the self-assessment. Since many of the people interviewed had experience with internal appraisals, where they may speak with people outside of their business unit or external people, the cooperative appraisal team did not experience any difficulties in this area.

Internal process improvement requires senior management commitment. The cooperative appraisal teams usually are comprised of several process group members. These process group personnel are responsible for recommending the future direction of process improvement efforts in the organization. The organizations being appraised were transitioning from CMM to CMMI; some had never piloted a CMMI appraisal. This provided an opportunity for the process group to gather valuable information on the state of the organization and how successfully the transition efforts were proceeding. The external inputs from the government can assist the process group by providing an unbiased view of the organization, and help influence senior management in determining what areas the organization should be concentrating on in their future process improvement initiatives.

"... the authors can state that an extremely beneficial byproduct of being on an appraisal team is that you can learn of many practices that can be adapted for use in your organization."

The government gained insight into the different cultures of the organizations comprising the contractor team. They had different ways of doing business, different vocabularies, and different ways of working with the prime contractor and the government. Understanding these differences aids in facilitating communications among the members of the IPTs and avoids misinterpretations.

Having a core team of government participants on all of the appraisals provided continuity. The same personnel can compare and contrast how the different organizations are performing process improvement and satisfying the practices of the CMMI. Common areas of weakness and interpretation issues can be raised and addressed to better support the program.

If the government provides several people for a cooperative appraisal, they should represent the major disciplines (e.g., program management, systems engineering, software engineering, and logistics) that are involved in the acquisition. A multi-disciplinary team can cover a broader range of PAs, while a given government team member can specialize in his or her area of expertise. Involving multiple disciplines allows the government to examine the developers' processes, taking into consideration the entire program life cycle. This enables downstream risks to be identified early in the acquisition. These risks can be mitigated with less effort as a result of the early collaboration and expertise among the government appraisal team and the developers' project teams.

PMs considering using cooperative appraisals should enlist direct or matrix support from the PM office for participation on the appraisal team. The detailed view of the organization is extremely valuable to a person who is supporting the PM in managing the acquisition. The appraisal team member gets to meet and speak with many of the people performing the project work, interview senior management, and understand their commitment to the project.

As participants of several appraisals, the authors can state that an extremely beneficial byproduct of being on an appraisal team is that you can learn of many practices that can be adapted for use in your organization. No organization has a monopoly on best practices. The more exposure that you have to the industry, the more you can recognize there are superior ways of doing things.

Aftermath

The joint appraisals were considered to be a success by both the government and industry. Two of the teammates who had been appraised invited the government representatives back for a follow-on appraisal where the company was trying to achieve a higher maturity level. These organizations valued the government's contributions to the appraisal effort, and wished to maintain the same experienced

appraisal team in this follow-on appraisal. The government, in turn, appreciated the opportunity to follow up on re-appraising the organizations. This allowed the government the ability to witness the effect of the process improvements made in the organizations since the self-assessment.◆

References

1. Members of the Assessment Method Integrated Team. Standard CMMI Appraisal Method for Process Improvement, Vers. 1.1: Method Definition Document. Pittsburgh, PA: Software Engineering Institute, Dec. 2001 <www.sei.cmu.edu/publications/documents/01.reports/01hb001.html>.
2. Office of the Secretary of Defense. Interim Defense Acquisition Guidebook. Washington, D.C.: Department of Defense, 30 Oct. 2002 <http://dod5000.dau.mil/DoD5000Interactive/InterimGuidebook.asp>.
3. Air Force Materiel Command. "Software Development Capability Evaluation." AFMC Pamphlet 63-103. Washington, D.C.: Department of the Air Force, 15 June 1994 <http://afmc.wpafb.af.mil/pdl/afmc/63afmc.htm>.
4. Members of the Assessment Method Integrated Team. Standard CMMI Appraisal Method for Process Improvement, Vers. 1.1: Method Implementation Guidance for Government Source Selection and Contract Process Monitoring. Pittsburgh, PA: Software Engineering Institute, Sept. 2002 <www.sei.cmu.edu/pub/documents/02.reports/pdf/02hb002.pdf>.
5. Phillips, Mike. "CMMI: A Progress Report." news@sei. Apr. 2005 <www.sei.cmu.edu/news-at-sei>.
6. National Defense Industrial Association. Guidebook and Training Breakout Group Workshop on CMMI Use in Acquisition. Proc. of CMMI Use in DoD Programs Workshop and Summit, Alexandria, VA., Sept. 7-8, 2005 <http://proceedings.ndia.org/587J/Gb_workshop.pdf>.

About the Authors

Diane A. Glaser is a computer scientist for the U.S. Army Communications-Electronics Life Cycle Management Command Software Engineering Center (SEC) at Fort Monmouth, N.J. As a systems analyst, she performed software design, development, and integration for communications systems. Glaser has participated in several appraisals for the government using the Standard Capability Maturity Model® Integration (CMMI®) Appraisal Method for Process Improvement, and has served on both government-only and cooperative government/industry teams. She belongs to the SEC CMMI Process Group and the SEC Software Engineering Process Group for the Battlespace Systems Support Directorate. Glaser has a Bachelor of Science in computer science from Montclair State University, New Jersey.

U.S. Army C-E LCMC
Software Engineering Center
BLDG 1210 RM 328
Fort Monmouth, NJ 07703
Phone: (732) 532-3287
DSN: 992-3287
E-mail: diane.glaser@us.army.mil

Michael D. Barnett is the Capability Maturity Model® Integration coordinator at MTC Technologies, provider of a wide range of sophisticated system engineering, intelligence, information technology, and program management solutions, primarily to the Department of Defense and various intelligence agencies. He has more than 25 years experience in developing and monitoring software-intensive systems, and has participated in several appraisals, both internal and external. He has Bachelor of Arts degrees in physics and astronomy from the University of Virginia and a Master of Science in computer science from Stevens Institute of Technology.

MTC Technologies
Information Dominance Division
25 James WAY
Eatontown, NJ 07724
Phone: (732) 440-1139
Fax: (732) 389-8708
E-mail: michael.barnett@mtctechnologies.com

The Qualification of Software Development Tools From the DO-178B Certification Perspective

Dr. Andrew J. Kornecki
Embry Riddle Aeronautical University

Dr. Janusz Zalewski
Florida Gulf Coast University

Software development tools are in wide use among safety-critical system developers. Examples of such use include aviation, automotive, space, nuclear, railroad, medical, and military applications. However, verification of tool output to ensure safety, mandated in highly regulated industries, requires enormous effort. If a tool is qualified, this effort can be reduced or even eliminated. The Radio Technical Commission for Aeronautics Document Order-178B and related documents provide guidelines by which to qualify these tools. However, current regulations, business models, and industry practice make this goal difficult to accomplish. This article discusses the qualification of development tools and the potential impact of this process on the aviation industry.

Software development tools are computer programs that help developers create other programs. Such tools have been in use since the early days of computing to improve the efficiency of the development process by automating mundane translation operations and bringing the level of abstraction closer to the application engineer. Nowadays, development tools are used in a variety of safety-critical applications, including the aviation, automotive, space, nuclear, railroad, medical, and military industries, and contribute to the risks associated with using respective products. Despite these risks to society, development tools are rarely qualified in a sense comparable to product certification in regulated industries. The objective of this article is to look at the current state of the tool qualification process, identify the issues, and propose recommendations for potential improvement, focusing on the aviation industry.

System Certification Versus Software Tool Qualification

Certification of airborne equipment is typically achieved through the Federal Aviation Administration (FAA) authorization of a type certificate (the entire aircraft), supplemental type certificate (new equipment in a specific aircraft), or a technical standard order (minimum performance standard for materials, parts, and appliances used on civil aircraft). A special committee (SC-145) of the Radio Technical Commission for Aeronautics (RTCA) convened in 1980 to establish guidelines for developing airborne systems. The report "Software Considerations in Airborne Systems and Equipment Certification" was published in January 1982 as the RTCA Document Order (DO)-178 (and revised as DO-178A in 1985).

Due to rapid advances in technology, the RTCA established a new committee (SC-167) in 1989 with the objective of updating the DO-178A by focusing on five areas: documentation integration and production, system issues, software development, software verification, and software configuration management and software quality assurance. The resulting document, DO-178B, provides guidelines for applicants developing software-intensive airborne systems [1, 2]. It discusses objectives that need to be met to show that the software development process provides specified levels of safety assurance. It also describes the processes and means of compliance.

Systems are categorized by DO-178B as meeting safety assurance levels A through E based on their criticality in supporting safe aircraft flight. The level A system is the most critical: The failure of such a system could result in a catastrophic failure condition for the aircraft. The level E system is the least critical: Such a system has no effect on the operational capability of the aircraft or pilot workload. Although the RTCA DO-178B is the leading source of guidelines for software developers engaged in such system construction, two other documents have critical bearing on the subject. RTCA DO-248B [3] clarifies some of the misinterpretation of the DO-178B. The FAA Order 8110.49 compiles a variety of guidelines related to the use of software in airborne systems. Chapter 9 is specifically dedicated to tool qualification [4].

A key component of the updated version of DO-178B is the concept of tool qualification elaborated in Section 12. Qualification is a supplementary process that the applicant may elect to follow in the course of certifying an airborne system. According to the definition given in DO-178B, tool qualification is defined as, "The process necessary to obtain certification credit for a software tool within the context of a specific airborne system." It is the certification authority that decides on the outcome of the qualification process. Moreover, qualification, if claimed, is a requirement in getting a system certified.

Types of Software Development Tools

DO-178B differentiates between verification tools *that cannot introduce errors but may fail to detect them* and development tools *whose output is part of airborne software and thus can introduce errors*. There is a significant amount of effort involved to qualify a verification tool, and much more to qualify a development tool. However, numerous development tools have been used successfully in many certified projects without being qualified. To define a subject matter more narrowly, we need to take a closer look at the entire domain of software development tools.

The landscape of modern software development tools is very broad, as illustrated in Figure 1 (see page 20). Following the traditional model of the development process from requirements to implementation, we can identify the following:

- The *requirements* category that includes tools used early in the life cycle to identify and specify the software requirements.
- The *design* category that includes tools allowing developers to create architectural and detailed design of the software in a notation of their choice supported by the tool; often in this category, tools translate the model to source code.
- The *implementation* category that includes all support required to translate the computer code and transfer it to the target computer.

As illustrated in Figure 1, three other categories of tools can be identified: those related to *analysis*, *testing*, and *target*.

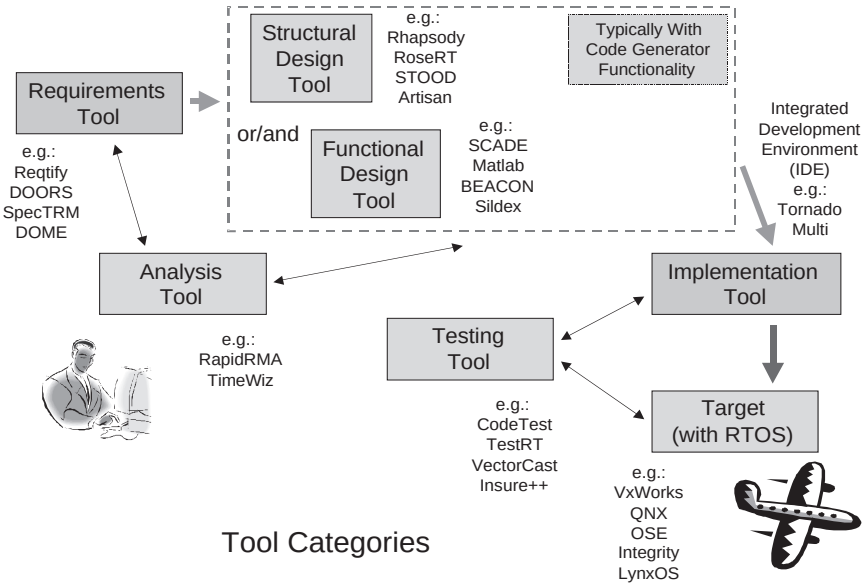


Figure 1: *Software Tool Categories*

However, for this article, we will focus primarily on the tools used in the design phase, the central component of the software development life cycle. They reflect two diverse viewpoints on real-time, safety-critical systems development, which result from different developers' backgrounds:

- Control engineers consider a system to be a dynamic model consisting of well-defined blocks of specific functionality (logic, arithmetic, dynamic). The functional paradigm of the model is the basis for system simulation and analysis of its behavior. Subsequently, the model can be translated *automatically* into an equivalent code, typically without any additional developer's involvement.
- Software engineers, on the other hand, are familiar with the concepts of operating systems, programming languages, software development methodologies, and notations. The graphic notations (classes, packages, states, transitions, events) allow developers to represent the structure and behavior of the target system software as a set of components that can be translated into programming constructs (data structures, objects, functions, etc.) using the automatic code generation functionality of the tool.

Consequently, the software design tools, which assist developers in translating the software requirements into source code, can be categorized into two groups: (a) a *function-based*, *block-oriented* approach applied by control and system engineers, and (b) a *structure-based*, *object-oriented* approach applied by computer scientists and software engineers.

The Qualification Process

A typical use for a design tool (software *producer*) is to transform an input artifact into output, thus creating another software artifact. The current process mandates verification after each transformation. If this transformation has an impact on the final airborne product, the producer needs to be qualified, but only if the transformation output would not be verified and the transformation leads to elimination, reduction, or automation of any of the DO-178B processes. The conditions under which a development tool requires qualification are presented in Figure 2 [4].

Software development tool qualification is attempted only as an integral component of a specific application program requiring the FAA's certification. The software tools to be used are referenced within the Plan for Software Aspects of Certification (PSAC) and the Software Accomplishment Summary documents of the original certification project. If development tool qualification is required, the applicant should present for review the Tool Operational Requirements (TOR) – a document describing tool functionality, environment, installation, operation manual, development process, and expected responses (also in abnormal conditions).

Two documents must be submitted and approved: a Tool Qualification Plan, and a Tool Accomplishment Summary as described in [4]. To make an argument for qualification, the applicant must demonstrate correctness, consistency, and completeness of the TOR and show that the tool complies with its TOR. This demonstration may involve a trial period during which a verification of the tool output is performed and tool-related problems are analyzed, recorded, and corrected.

Other data required for review include a Tool Configuration Management Index, Tool Development Data, Tool Verification Records, Tool Quality Assurance Records, Tool Configuration Management Records, etc. These requirements are also described in [4]. Tool qualification data are approved only in the context of the overall software development for the specific system where the intention to use the tool is stated in the PSAC. The tool itself does not receive a separate *qualification stamp of approval*. Therefore, using the tool on another system/project requires a separate qualification, although some qualification credits may be reused.

Surveys requesting which tools are used by industry were conducted at two national conferences: the 2002 FAA National Software Conference and the 2004 Embry Riddle Aeronautical University/FAA Software Tool Forum. In addition, two follow-up e-mail solicitations were sent to more than 500 professionals working on airborne systems. These surveys and solicitations resulted in a relatively small sample of responses that did not provide a base for statistically significant results. The comments included industry discouragement regarding the rigor of development tool qualification, and a justified perception of the extensive cost of qualification.

Potential solutions to assist in commercial off-the-shelf (COTS) development tool qualification included extensive vendor collaboration and using *alternate means* allowed in DO-178B. The limited feedback shows that there has been interest in qualifying software development tools classified in the function-based/block-oriented category, which cannot be said about structure-based/object-oriented tools.

A short list of qualified development tools includes code generators (Generation Automatique de Logiciel Avionique, Graphical Processing Utility, Virtual Application Prototyping System Code Generator, Safety Critical Avionic Development Environment Qualifiable Code Generator) and configuration-scheduling table generators (Universal Table Builder Tool, Configuration Table Generation Tool), most of them being in-house products. According to several informal exchanges with industry, many of the modern COTS software development suites actually have been used in the creation of software artifacts on certified projects without going through the qualification process.

Problems With Development Tool Qualification

It is clear that qualification of develop-

ment tools is an option rarely exercised in the airborne software industry. In fact, one could argue that qualification of development tools is not a viable option. Current interpretation of applicable guidelines makes development tool qualification a proposition that is not practical from a *managerial viewpoint*, and not easy from a *technical viewpoint*.

Managerial Viewpoint

The first group of problems is of a regulatory and managerial nature. The major hurdle is the current state of regulations and guidelines. The secondary obstacle is the business model and lack of incentives, in particular the prohibitive cost of tool qualification. The existing tools, often used in certification projects, do not have appropriate data to support arguments about meeting the objectives of DO-178B. The applicant team's intent is to certify the product rather than expand effort and qualify the tool. The tool vendor does not see the business advantage of qualifying a tool while disclosing proprietary information to potential competitors.

Development tool qualification requires close collaboration between the tool vendor and the applicant. This is the reason why in-house tools are more likely to be qualified. Internal trade studies [5] have shown that the cost of development tool qualification is significantly higher than the cost of verification tool qualification. The use of qualified verification tools can result in fast savings on the first program where they are introduced. In contrast, the use of qualified development tools may require several programs to make up the cost.

The intellectual property rights may need to be waived by the vendor to achieve qualification. The tool cannot be qualified as standalone, but only within the scope of a particular certification project. The tools that could be considered for qualification are very simple: typically in-house created utilities where the applicant holds all intellectual property rights, maintains all tool development data, and can reuse the tool software artifacts on consecutive projects. The qualification is accomplished within the specific certification project and thus is not clearly visible from the outside as *development tool qualification*.

Technical Viewpoint

The second group of problems is related to technical aspects. According to the DO-178B interpretation, the development tool needs to be qualified to the same level of scrutiny as the appropriate application it is helping to develop. However, there is a significant difference between tool software and application software. Applications run on a target computer while tools operate on a general-purpose workstation, typically closely interacting with a COTS operating system and conventional programming environment. Considering this, several DO-178B objectives are not applicable to tool software and thus cannot be met. There is also no general agreement on what metrics would allow developers to carry an independent tool assessment [6].

One often-repeated statement regarding development tool qualification is the requirement that "only deterministic tools can be qualified." The DO-178B refers to determinism as "… tools which produce the same output for the same input data when operating in the same environment." The definition does not take into account how the output is generated. By this definition, one may interpret that it is not required to provide proof on the internal behavior of a tool. An example of this can be memory use for a tool running on the host workstation in a multitasking, multiuser, networked environment. The problem is to define what the object code for a tool is. Does it include the operating system (OS) of the host workstation? A tool clearly needs to make explicit calls to the OS routines, and any verification of these would require full visibility of the host's OS and related high assurance of its operation.

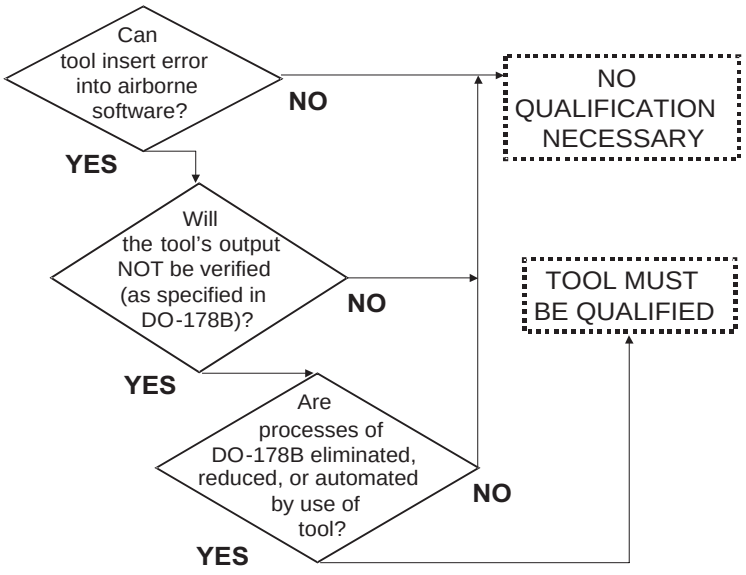
The main function of a software development tool is to transform, i.e., translate an input artifact into output. This is why the qualification, if applicable, should be focused on this translation component of the tool functionality. However, modern, complex software development tools provide a variety of other functions that are not directly related to the translation process. The translation component is hidden deep inside the tool, which causes problems with tool qualification. Typically, there is no access to a COTS tool's life-cycle data, which describe the tool's requirements, design, and code. Unless the tool has been developed in-house, the qualification efforts may be doomed.

Potential Solutions

The qualification of a stand-alone development tool is not feasible in the strict sense of existing guidelines. Such concepts as component-based software, software reuse, and service history should be explored [7] to identify the feasibility of such qualification. The issues of tool version control and the precise definition of operational environment, constraints, and limitations are the basis for starting discussion about solutions to tool qualification. The availability of extensive tool software development data, often scarce for COTS products, may be a challenge to ever accomplish COTS tool qualification [8].

It could be conceivable to create an independent lab dedicated to tool qualification and encourage commercial vendors to submit their product for assessment. A similar approach is known from other areas of verification and validation [9, 10]. Another idea would be to require certified product applicants to disclose information regarding the development tool use and qualification effort by creating an FAA-sponsored database for DO-178B certified products. This could face serious objections from industry due to an apprehensiveness to disclose any information, which may result in the loss of commercial advantage. It would be possible to research a potential for development tool

Figure 2: *Conditions When a Software Development Tool Requires Qualification*



qualification using an approach different than the one outlined in Section 12.2 of DO-178B. Service history and formal methods could both be potential options.

It appears that the industry has a pressing need to come up with methods to audit a tool that is independent of the specific program and applications using it. This would require updating the guidelines to consider a model-driven development paradigm, redefine the qualification process, and allow flexibility regarding qualification to be less dependent on the application program using the tool. A more streamlined method to qualify development tools and to keep them current as technology advances would be useful. Better guidance on how to apply service history and how to address what has to be done for incremental tool changes would also be needed. These and other issues have been discussed at the recent Tools Forum [11]. The RTCA convened another special committee (SC-205) with a charge to recommend modifications to the existing DO-178B. The qualification of software tools is being discussed and some changes may be forthcoming.◆

Acknowledgement

The presented work was supported in part by the Aviation Airworthiness Center of Excellence under contract DTFA-0301 C00048 sponsored by the FAA. Findings contained herein are not necessarily those of the FAA. Additional support was received from the Florida Space Grant Consortium under Grant No. UCF01-E000029751.

References

1. Radio Technical Commission for Aeronautics, Inc. "RTCA DO-178B, Software Considerations in Airborne Systems and Equipment Certification." Advisory Circular. Washington, D.C.: RTCA, 1 Dec. 1992 <www.rtca.org/downloads/ListOfAvailableDocsAPR%202005.htm#_Toc101071800>.
2. Federal Aviation Administration. "RTCA Inc., Document RTCA/DO-178B." Advisory Circular No. 20-115B. Washington, D.C.: U.S. Department of Transportation, Nov. 1993 <www.airweb.faa.gov/Regulatory_and_Guidance_Library/rgAdvisoryCircular.nsf/0/DCDB1D2031B19791862569AE007833E7?OpenDocument>.
3. Radio Technical Commission for Aeronautics, Inc. "RTCA DO-248B, Final Report for Clarification of DO-178B 'Software Considerations in Airborne Systems and Equipment Certification'." Advisory Circular. Washington, D.C.: RTCA, 10 Dec. 2001 <www.rtca.org/downloads/ListOfAvailableDocsAPR%202005.htm#_Toc101071717>.
4. Federal Aviation Administration. "Software Approval Guidelines." FAA Order 8110.49. Washington, D.C.: FAA, 2003 (Chapter 9 replaces FAA Notice N8110.91 of 2001) <www.airweb.faa.gov/Regulatory_and_Guidance_Library/rgOrders.nsf/0/640711B7B75DD3D486256D3C006F034F?OpenDocument&Highlight=8110.49>.
5. Potter, Bill. "Use of the MathWorks Tool Suite to Develop DO-178B Certified Code." Slide No. 13. Honeywell, May 2004 <http://faculty.erau.edu/korn/ToolForum/potter.htm>.
6. Kornecki A., and J. Zalewski. Criteria for Software Tools Evaluation in the Development of Safety-Critical Real-Time Systems. Proc. of PSAM-7/ European Safety and Reliability Conference, Berlin, Germany, 14-18 June 2004. London: Springer-Verlag, 2004 <http://faculty.erau.edu/korn/papers/ESREL04KorneckiZalewski.pdf>.
7. Lougee, H. "DO-178B Certified Software: A Formal Reuse Analysis Approach." CrossTalk Jan. 2005 <www.stsc.hill.af.mil/crosstalk/2005/01/0501lougee.html>.
8. Zalewski, J., W. Ehrenberger, F. Saglietti, J. Gorski, and A. Kornecki. "Safety of Computer Control Systems: Challenges and Results in Software Development." Annual Reviews in Control 27.1 (2003): 23-37.
9. Brosgol, B.M. "ADA in the 21st Century." CrossTalk Mar. 2001 <www.stsc.hill.af.mil/crosstalk/2001/03/brosgol.html>.
10. Adams, M., et al. "Conformance Testing of VMEbus and Multibus II Products." Advanced Multi-Microprocessor Bus Architectures. Ed. J. Zalewski. Los Alamitos, CA.: IEEE Computer Society Press, 1995: 392-399.
11. Embry Riddle Aeronautical University/FAA Software Tools Forum, Embry Riddle Aeronautical University, Daytona Beach, FL., May 18-19, 2004 <www.erau.edu/db/campus/softwaretoolsforum.html>.

About the Authors

Andrew J. Kornecki, Ph.D., is a professor at the Department of Computer and Software Engineering, Embry Riddle Aeronautical University. He has more than 20 years of research and teaching experience in areas of real-time computer systems. Kornecki contributed to research on intelligent simulation training systems, safety-critical software systems, and served as a visiting researcher with the Federal Aviation Administration (FAA). He has been conducting industrial training on real-time, safety-critical software in medical and aviation industries and for the FAA Certification Services. Recently, he has been engaged in work on certification issues and assessment of development tools for real-time, safety-critical systems.

Dept. of Computer and Software Engineering
Embry Riddle Aeronautical University
600 Clyde Morris BLVD
Daytona Beach, FL 32114
Phone: (386) 226-6888
Fax: (386) 226-6678
E-mail: kornecka@erau.edu

Janusz Zalewski, Ph.D., is a professor of computer science at Florida Gulf Coast University. Prior to this, he worked for various nuclear research institutions, including the Data Acquisition Group of Superconducting Super Collider and Computer Safety and Reliability Center at Lawrence Livermore National Laboratory. He also worked on projects and consulted for a number of private companies, including Lockheed Martin, Harris, and Boeing. Zalewski served as a chairman of the International Federation for Information Processing Working Group 5.4 on Industrial Software Quality, and of an International Federation of Automatic Control Technical Committee on Safety of Computer Control Systems. His major research interests include safety-related, real-time computer systems.

Dept. of Computer Science
Florida Gulf Coast University
10501 FGCU BLVD
Fort Myers, FL 33965
Phone: (239) 590-7317
Fax: (239) 590-7330
E-mail: zalewski@fgcu.edu

Using Line of Balance to Track the Progress of Fixing Trouble Reports

Eduardo Miranda
Independent Consultant

You are the project manager of a large project and testing is uncovering faults, trouble reports are starting to pile up and the release date is coming soon. Are they going to be fixed on time? What could you do to help? Are there any bottlenecks? Where should you assign more resources? Does this scenario sound familiar? Have you been there? This article will explain how you can answer these questions by using an old method called Line of Balance in a new way.

Line of balance (LOB) was devised by the members of a group headed by George E. Fouch during the 1940's to monitor production at the Goodyear Tire & Rubber Company [1]. It was also successfully applied to the production planning and scheduling of the huge Navy mobilization program of World War II and during the Korean hostilities. Today, LOB application has been further expanded, making it suitable for a whole spectrum of activities ranging from research and development through job-shop and process flow operations.

In the context of managing a software project, the LOB technique offers two main advantages over the traditional *Open Trouble Reports (TRs) Chart* [2]:

- It allows project managers to see, in the middle of a project, whether they can meet the schedule if they continue working as they have been.
- It exposes process bottlenecks, allowing the project manager to focus on those points responsible for slippage.

The Open TRs Chart

To answer some of the questions raised at the beginning of this article, project managers usually resort to the Open TRs Chart shown in Figure 1 or a variation of it.

The Open TRs Chart shows the cumulative number of TRs written over time, and its breakdown into open and closed TRs. As the project progresses, the closed line should converge toward the total line and the open line towards zero. A *closed* line that is not converging fast enough toward the total or an *open* line that does not approach zero signals to the project manager the need to devote additional resources to fix problems.

Variations of the chart include showing a more detailed breakdown of the TR status, and ratios between total and open TRs [2, 3].

Despite all its usefulness, the Open TR Chart lacks predictive ability and fails to take advantage of past and present performance data and TRs closure targets; i.e., how many TRs should be in a given state by a given time to meet the project deadlines. In other words, although the chart will give the project manager a gut feeling about the situation, it would not answer the questions of where are we in relation to where we are suppose to be, or how much better we should be doing to get where we want to get by the time we want.

The TR Life Cycle

Typically, a TR will go through a number of stages or states since it is reported until it is closed (see Figure 2, page 24). Each of these states corresponds to a milestone in the process of answering a TR into which the organization or project manager wants to have visibility to evaluate progress, i.e., how many TRs have been reported, how many of the reported TRs have been analyzed, how many of the analyzed were rejected and so on. Elemental states could be grouped into super sets for reporting purposes, i.e., while the project manager might be interested in how many have been analyzed, assigned, implemented, or integrated the steering group overseeing the project might only been interested in how many TRs were reported, how many were closed, and how many were still pending.

Most defect tracking systems will implement this model or some variation of it, time stamping each TR as they transition between states. This last feature would allow the organization to produce the lead-time information required by the LOB method.

In addition to the state and timing information, the TR includes other data such as the severity of the problem. This information could be used to filter the TR data and apply the LOB method to a subset of all the TRs reported and in the prioritization of which TRs to fix first.

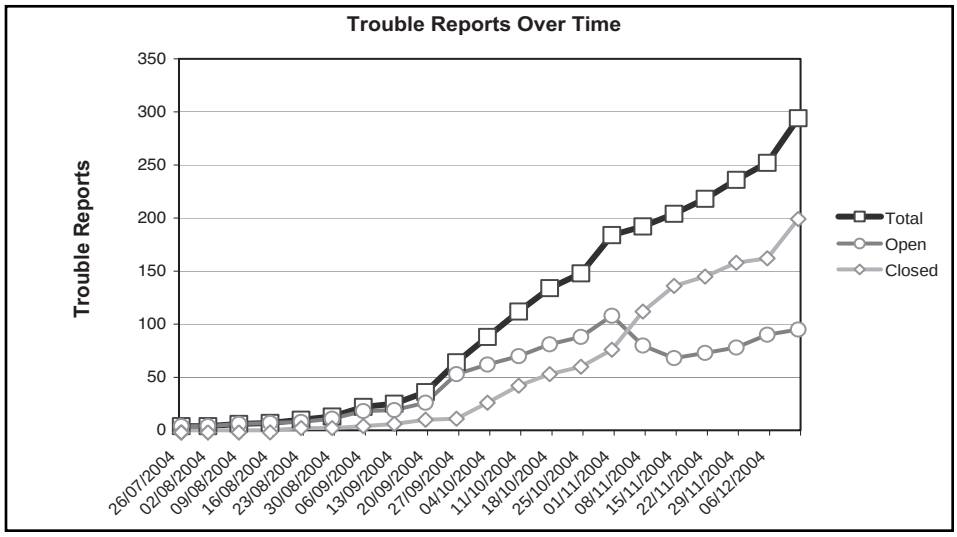
The LOB Method Applied to TRs

The LOB method consists of the following elements [4]:

- A number of control points and their lead times to closing as illustrated in Figure 3 (see page 24), at which progress is to be monitored.
- An *Objective* Chart or target plan displaying the cumulative closing schedule as planned by the project manager to meet a set deadline (Figure 4).
- The TRs Status Chart (see Figure 5, page 25), which shows the actual number of TRs that have passed through a given control point versus the number that should have been passed (the LOB) according to the plan.

The information contained in the

Figure 1: *Open Trouble Reports Chart*



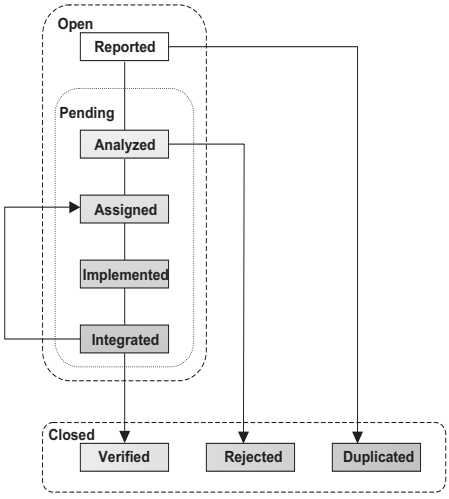


Figure 2: *Typical TR Life Cycle*

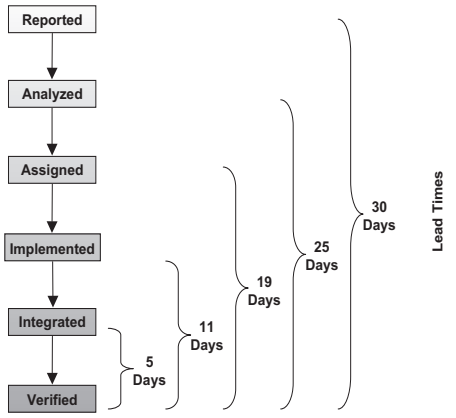


Figure 3: *The Process of Solving a TR and Its Corresponding Lead Times*

Control Point	Time in State	Lead Time
Reported	5	5 + 25 = 30
Analyzed	6	6 + 19 = 25
Assigned	8	8 + 11 = 19
Implemented	6	6 + 5 = 11
Integrated	5	5 + 0 = 5
Verified	0	0

Table 1: *Lead-Time Calculations*

Objective Chart, together with the lead-time information is used to calculate how many TRs should be in a given state at a given time.

Control Points

In LOB terminology, a control point is a milestone or event that the project manager wants to monitor. In the context of tracking TRs, the control points and states in the TR life cycle would most likely coincide, but this is not necessary[1]. For example, the project manager might not find it useful to track TRs in the rejected state and so this state would not be considered a control point. The lead time for a control point is calculated using the following formula:

$$LeadTime_n = 0$$

$$LeadTime_{q=n-1,n-2,..,n-1} = TimeInState_q + LeadTime_{q+1}$$

Assuming that the median[2] times a TR spends in a given state are those shown in Table 1, the lead-time calculations will yield the results illustrated in Figure 3.

The Objective Chart

The Objective Chart shows cumulative, *to be verified* TRs on the vertical scale and dates of achievement along the horizontal scale. The chart might also include a display of the achievements so far.

The Objective Chart in Figure 4 shows that the project manager has committed to close 50 TRs by the end of September, 80 by the end of November, and 150 by the beginning of the following year. The chart also shows that as of mid-December progress is slightly behind with the project delivering around 75 fixed TRs instead of the 80 promised.

TR Status Chart

The TR Status Chart provides quantitative information with regards to progress, and whether or not there is a bottleneck on the process.

The chart portrays the actual number of TRs that have passed through each control point against the number that should have been passed according to the plan. These last quantities are called the LOB. The difference between the LOB and the top of the bar for each control point is the number of TRs behind or ahead of schedule.

Notice that the shape of the LOB will change daily even if there are no new TRs reported, since its calculation depends on the planned curve of the Objective Chart and the status date.

The TR Status Chart shows that there are almost 180 TRs reported so far, 30 more than what were planned to fix according to the Objective Chart. This signals the need to update the plan. It also tells us that TR implementation is on track as the actual column and the LOB line for that control point coincide, but that we are falling behind in their integration and verification. This suggests that adding more people to implementation activities will not help recoup the delay, but that additional resources could be used in integration and verification activities.

The LOB for each control point is calculated as follows:

$$LOB_{q=1,2,..,n} \begin{cases} a_1 + b_1 t & t_1 \le t < t_2 \\ a_2 + b_2 t & t_2 \le t < t_3 \\ \vdots & \\ a_m + b_m t & t_m \le t < t_{m+1} \end{cases}$$

$$b_i = \frac{y_{i+1} - y_i}{t_{i+1} - t_i}$$

y_i and y_{i+1} are the number of TRs to be fixed by time t_i and t_{i+1} respectively, as planned by the project manager and captured in the objective chart.

$$a_i = y_i - b_i t_i$$

$$t = TimeNow + LeadTime_q$$

The idea behind the procedure is simple. If it takes an average of 10 days for a TR to go from a given state to the completion state, today's status for that state should be equal to the number of TRs that would have to be completed according to the plan 10 days from now. See Figure 6 for a graphical example.

In Figure 6, the chart on the left shows the planned line from Figure 4, while the chart on the right shows the scheduled

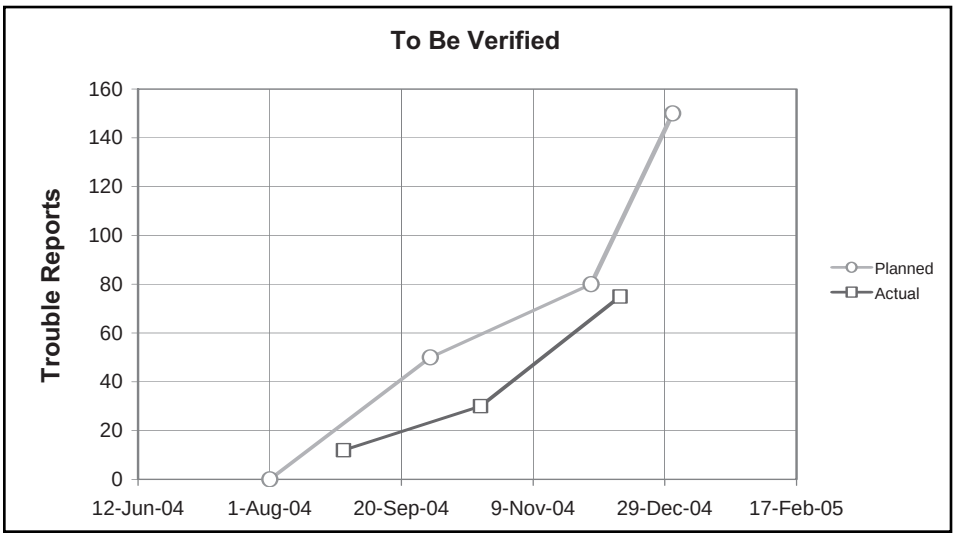


Figure 4: *The Plan Proposed by the PM to Clear the TR Backlog*

line from Figure 5. We obtained Figure 5's scheduled line by finding the interception between the *TimeNow* line and the curve in the objective chart (the $a_1 + b_2 t, \ldots, a_m + b_m t$ function above), which yields the value for the Verified Control Point, that is the number of TRs that should be on that state as of Dec. 12, 2004. The interception between the curve and the line at $TimeNow + LeadTime_{Implemented}$ yields the LOB value for the Implemented Control Point.

Summary

By providing a credible early warning about bottlenecks in the process of fixing TRs, the LOB method helps project managers take corrective actions such as allocating more resources or prioritizing the work when there is still time to do it.

In terms of the data required to implement the LOB technique, most of it should be readily available from your defect tracking system or could be derived from it with a few calculations implemented in Excel or any other spreadsheet.◆

Acknowledgements

Thanks to Jeremy O'Sullivan and Gaetano Lombardi from Ericsson; Alain Abran from École de Technologie Supérieure - Université du Québec; and Raul Martinez from RMyA for their comments on earlier versions of this article; and to John Corcoran from Ericsson for the TR statistics.

References

1. Harroff, Noel N. "Line of Balance." NNH Enterprise, 7 June 2003 <www.nnh.com>.
2. Pussacq Laborde, Juan. "Quality Control = Project Control?" Second Software Engineering Process Group Latino America Conference. Mexico, 2005 <www.esi.es/SEPGLA/index_eng.html>.
3. Florac, William A. "Software Quality Measurement: A Framework for Counting Problems and Defects." CMU/SEI-92-TR-22. Pittsburgh, PA: Software Engineering Institute, 1992.
4. Miranda, Eduardo. Running The Successful Hi-Tech Project Office. Artech House, 2003.
5. Defense Acquisition University. Scheduling for Program Managers. Defence Systems Management. College Press, Oct. 2001 <www.dau.mil/pubs/gdbks/scheduling_guide.asp>.

Notes

1. The control points are likely to be a subset of the TR states. To avoid confusion, do not create additional control points.

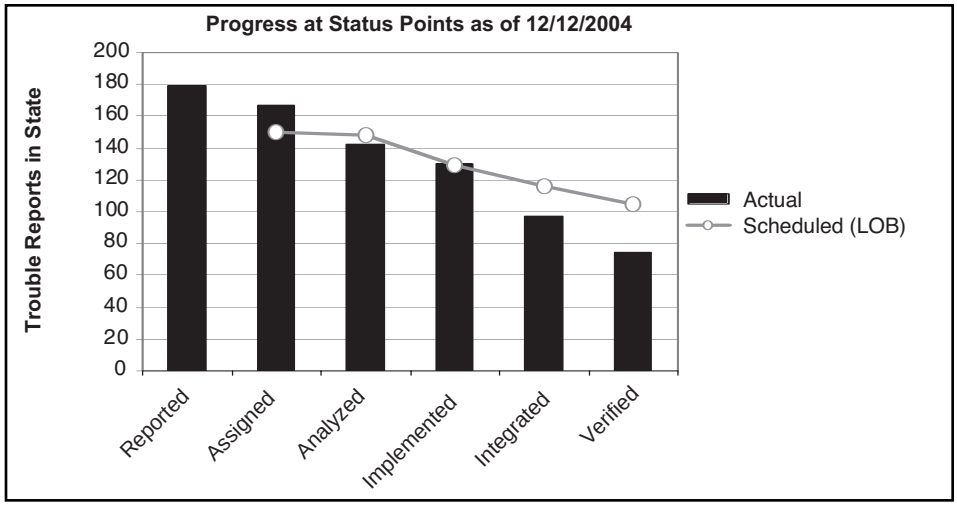


Figure 5: *Trouble Reports Status Chart*

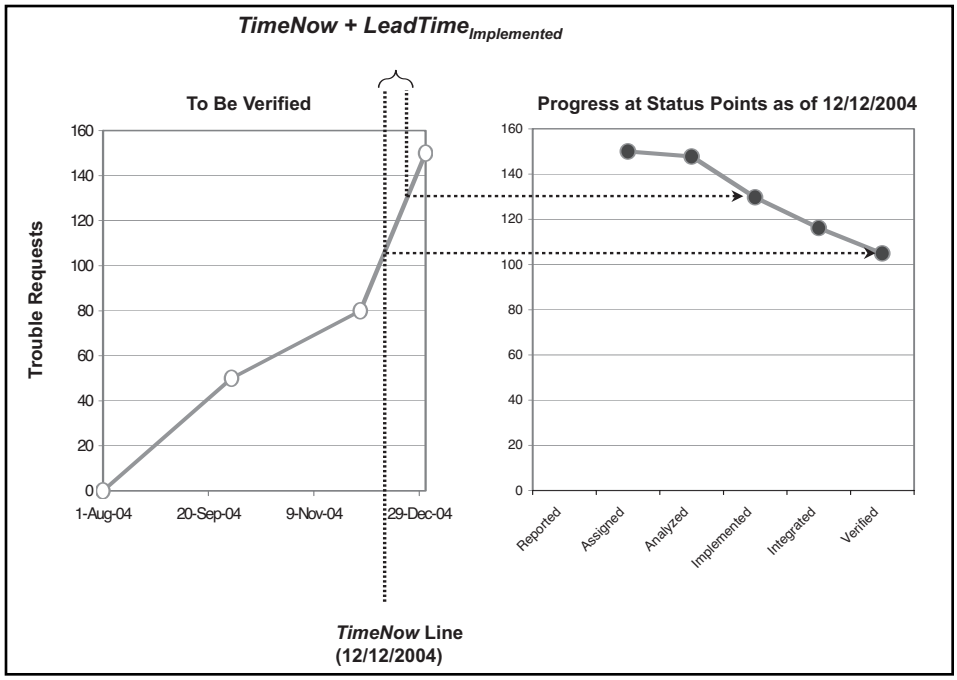


Figure 6: *The LOB for the Control Point*

2. The median is preferred to the arithmetic mean (average) to prevent rare but complex TRs from skewing the value of the statistic to the right.

About the Author

Eduardo Miranda is a system professional with 20 years of experience in the development of software-based products and information management systems. Currently, he works in the development of new estimation and planning approaches for research and development projects. Miranda is affiliated with the Université du Québec à Montréal as an industrial researcher, and is a member of the International Electronics and Engineers. He has published more than 10 papers in software development methodologies, estimation, and project management and is the author of "Running the Successful Hi-Tech Project Office." Miranda has a Master of Engineering degree from the University of Ottawa and a master's degree in project management from the University of Linkoping.

119 Harwood Gate
Beaconsfield, Quebec
Canada H9W 3A5
Phone: (514) 697-0594
E-mail: emt.miranda@computer.org

Letters to the Editor

Dear CrossTalk Editor,

In CrossTalk's February issue, Ken Schwaber commented on my December article, titled "Agile Software Development for the Entire Project." I am in firm agreement with him theoretically and philosophically. However, the open water reality of working in agile software development brings home how difficult agile software can sometimes become. In both thought and practice, it requires us to be open and willing to change to keep up with *evolution*. After all, if there is one thing that agile processes tell us, it is to embrace change. Perhaps it requires more change of the traditional mindset than one of the organization.

As such, the agile community has been working diligently to best understand how various roles – such as test – *fit* into agile software development processes. If one has the opportunity to attend an agile conference, one notices entire tracks devoted to testing the *results* of agile processes. Indeed, most organizations end up integrating testing on their own, using a *Chinese menu approach* [Bob Martin, keynote Agile Conference 2005] where they mix and match various agile practices to construct their own processes.

Working with on-site customers is always our best-case scenario. The feedback is direct, constant, and immediate. Realistically, however, many projects cannot bring customers on-site for perfectly good reasons. After all, customers have jobs too. What do we say to those projects? You can't be agile? No, we shift a little and bring in roles to go to work with the customers on their site. Customer validation is an extremely important part of any real-world agile process.

This article demonstrates the need for a next generation (XP is already on version 2.0) of agile processes with ways of dealing with customers who cannot spend time on-site, larger agile projects, distributed projects, and/or the need for testing of the delivered systems on behalf of the customer. You can consider any one of these conditions a barrier to using some of the first-generation processes. Of course, these new processes will have to continue to promote adaptive, self-managing teams in a team-oriented environment.

I wish to thank Mr. Schwaber for reading my article. I always appreciate his from-the-heart feedback. My goal is to strive to be true to agile methodology, while addressing the real-life ongoing, ever-changing, ever-evolving needs of our customers and our exciting industry.

Granville "Randy" Miller
Microsoft
<randymi@microsoft.com>

Dear CrossTalk Editor,

I very much liked the BackTalk column about software usability titled "Push for Cheese: A Metaphor for Software Usability" by Nicole Radziwell and Amy Shelton in the December 2005 issue of CrossTalk. It was very thought-provoking. I frequently get annoyed at software in which it is either too hard or too easy to invoke a selection or option, and it results in something I did not intend. Your suggestions for how to overcome this problem were *right on*. It is very challenging to have a user community that (a) is very diverse in its opinions on what makes for good software usability and, (b) as they get more familiar with the software, their preferences on how it should work change.

Thank you for drawing visibility to this seemingly simple but in actuality very complex issue. Hopefully it will prevent problems such as operator-machine issues coming up right before going to final system test that should have been addressed much earlier in the development.

Al Kaniss
Naval Air Systems Command
<alan.kaniss@navy.mil>

Web Sites

Tantara Inc.

www.tantara.ab.ca/info.htm

Tantara offers practical advice for software process improvement and software quality assurance/management, including the International Organization for Standardization (ISO) 9001 (ISO 9000-3, TickIT), ISO 12207, ISO 15504 (Software Process Improvement and Capability dEtermination [SPICE]), the Software Engineering Institute's Capability Maturity Model® (CMM®) for Software, CMM Integration℠, and more. Tantara's Web site is updated quarterly and offers articles (online, comparisons, frameworks); a bulletin board of events, news, and statistics; job aids; and books and training materials.

Software Process Improvement in Regions of Europe

www.cse.dcu.ie/spire

The Software Process Improvement in Regions of Europe (SPIRE) was a project funded by the European Commission involving partners in Austria, Ireland, Italy, Sweden, and Northern Ireland. SPIRE's objective is to help small software development units (SUDs) – employing a software staff no larger than 50 – to get business benefits from investment in the Software Process Improvement Network (SPIN) and to share their experiences with others. The project helped nearly 60 SUDs in SPIRE to carry out short, cost-effective improvement projects. Their experience has been captured in the European Analysis Report, which provides an analysis of the impact of SPIN within SUDs throughout Europe, and identifies trends and key features for future planning or implementation.

TickIT

www.tickit.org

TickIT guides the developer to achieve high-quality software within the framework of the ISO 9001. TickIT applies to all types of information systems that involve software development processes in the product life cycle. Typical systems suppliers include system houses, software houses, and in-house developers. TickIT disciplines are also relevant to the development of embedded software. A full definition of the scope of TickIT is included in the TickIT Guide, available on the Web site.

How to Relate Quality and Reuse in Evolving Systems

Dr. Ronald J. Leach
Howard University

This article suggests a model to predict the quality of software developed over time, where the reusable components are also evolving over time.

At NASA Goddard Space Flight Center, we analyzed several safety-critical software systems with sizes ranging from approximately 50,000 to more than 500,000 lines of code (LOC). These systems' architecture included a reusable software core linked to mission-specific software that resulted in complete, unique ground support systems for spacecraft control. Each spacecraft required both new, mission-specific source code and an interface to the reusable core. Because of technology changes, the reusable core itself evolved, increasing in size more than five-fold. The software systems have been somewhat superseded by commercial off-the-shelf products.

The evolving, reusable core had a much lower defect ratio (defects per thousand LOC [KLOC]) for the reusable core than similar systems in the same application domain. The defect ratios of 0.034 and 0.075 for the latest two versions of the reusable core discussed here are far lower than the range of 0.12 to 1.89 for similar systems. The extremely low defect ratio of the core was even more impressive in view of Les Hatton's statement that very few systems, even safety-critical ones, have ever stayed below one defect per KLOC [1].

The Model

What caused the unusually high quality of the evolving, reusable software core? Did the reused part of the core have a low defect ratio simply because it had been operational for so long in a safety-critical domain? If so, we should expect defects per KLOC to follow an exponential distribution of the form a constant times e^{K*T}.

On the other hand, was the new code good because it underwent stringent analysis and testing? If so, we should expect a linear relationship, reflecting the increase in code size.

These two questions led to a simple model for the number of defects in each release as the sum of an exponential distribution representing defects caused by reused code and a linear expression representing the defects in the new source code. The model is:

$$DEFECTS = (REUSED * e^{K*T} + NEW) * M * KLOC$$

where,

T is the time between releases. REUSED and NEW are the percentage of reused and new lines of code in a release, respectively.

KLOC is the size of the release.

The constants *K* and *M* are discussed later.

Calibration of the Model

Calibration of the model requires calculation of the constants *REUSED*, *NEW*, *K*, and *M*. We already know KLOC and want to estimate *DEFECTS*. The overall reuse percentages between releases clustered around 80; hence, the value of *NEW* was .20 and, thus, our the model became:

$$DEFECTS = (.80 * e^{K*T} + .20) * M * KLOC$$

To compute the constant *K*, we entered the defects and the KLOC for each release into an Excel spreadsheet, one entry per release, and then computed the exponent *K* in the exponential distribution, using the Excel LOGEST function to compute *K*. This constant was, to two decimal places, 0.79.

The constant *M* represents the change scale between LOC and *DEFECTS*. To compute *M*, we created a third column in the Excel spreadsheet, representing the difference between the observed number of defects for previous releases and what is predicted by the partial formula $(.80 * e^{-.79*T} + .20) * KLOC$, one entry per release, and then computed *M* using the Excel SLOPE function. This constant *M* was, to two decimal places, 0.65, making the model for this domain:

$$DEFECTS = (.80 * e^{-.79*T} + .20) * 0.65 * KLOC$$

Analysis and Future Work

It is possible to fine-tune the model more than what we presented here. Specific values can be used instead of the averages to improve accuracy. Using different multipliers of *REUSED* and *NEW* can improve estimation, also.

Measures based on exponential distributions frequently are used to assess system quality, reliability, and to stop testing when the expected number of errors remaining meets the objective error rate. As far as we know, their use in conjunction with software reuse to predict faults is new.

Note that applying reuse measurements to evolving code is somewhat controversial. Indeed, Jeffrey Poulin [2] and others recommend against attempting software reuse when the underlying code is not stable. The author recommends it in certain circumstances [3], while others are neutral or do not comment on the issue. Readers are encouraged to provide their experiences via e-mail to the author.◆

Acknowledgement

This research was partially supported by National Science Foundation grant number EIA-0324818.

References

1. Hatton, L. "Does OO Sync With What We Think?" IEEE Software 15.3 (May, 1998).
2. Poulin, J.S. Measuring Software Reuse: Principles, Practices, and Economic Models. Reading, MA: Addison-Wesley, 1997.
3. Leach, R.J. Software Reuse: Methods, Costs, and Models. New York: McGraw-Hill, 1996.

About the Author

Ronald J. Leach, Ph.D., is professor and chair of the department of systems and computer science at Howard University where he performs research on software engineering with special interest in reuse, metrics, and fault tolerance. He has a Bachelor of Science, Master of Science, and doctorate degree in mathematics from the University of Maryland, and a Master of Science in computer science from Johns Hopkins University.

Dept. of Systems and Computer Science
Howard University
Washington, DC 20059
Phone: (202) 806-6650
Fax: (202) 806-4531
E-mail: rjl@scs.howard.edu

When Did Six Sigma Stop Being a Statistical Measure?

Joe Schofield
Sandia National Laboratories

The term Six Sigma is widely used as an approach for process improvement and learning. It is a disciplined, structured, data-driven methodology to solving problems. Along the path to popularity, Six Sigma lost its meaning as a statistical measure and instead inherited the meaning of merely another measurement program. Organizations that intend to employ Six Sigma ought to consider which definition of six sigma *is their target: a process improvement approach, or a statistical measure for variation. This article explores the significance of the differences between six sigma and Six Sigma. Read on if you dare.*

When did *free* come to mean *free after rebate*? When did cost become *the China cost*? And when did *Six Sigma* become something other than a statistical notion? Or have you not noticed that the term *Six Sigma* no longer means a *statistical measure for variation*? For every organization that attempts to use six sigma as a statistical measure of process improvement, three other organizations use it merely to describe a process improvement effort. Most of these organizations have no intention of using six sigma statistically, but it likely impresses the folks higher up in the food chain.

Affixing lean to the term, as in Lean Six Sigma (LSS), is currently an institutional silver bullet. Do not feel left out if you have not been exposed to LSS; one Internet search found only about 125,000 LSS-related sites, but more than 1.7 million sites for Six Sigma. Very few of these sites advocate six sigma's statistical meaning, contributing to the miscommunication regarding Six Sigma processes.

As with most trendy initiatives, LSS has it own status symbols: green belts, black belts, and an assortment of colors and variations depending on the accrediting organization. In addition, LSS has a lexicon, words like kaizen, kaikaku, kanban (yes, there are more than just k words). There is one more Japanese word that the LSS industry may have forgotten: it is *muda,* or the word for *waste*. Without applying statistical measurement, organizations may be wasting their process improvement resources.

The application of LSS may bring numerous well-intended results, including defect reduction, work in progress reduction, cycle time reduction, cost savings, fewer hand-offs and queues, minimized changeover time, workload leveling, and more. Organizations in pursuit of process improvement are often well-advised to consider LSS to diagnose, improve, and measure their processes.

Motorola Corporation gets much of the credit for popularizing Six Sigma and the phrase *3.4 defects per million* – the battle cry of the Six Sigma world. Simply re-stated, Six Sigma has come to be synonymous with no more than 3.4 defects per million opportunities (DPMO). An opportunity might be defined as a keystroke or a mouse click, depending on whether the process being measured is developing software or writing an article.

Often, the value 3.4 DPMO is followed with a footnote or an asterisk; the fine print typically ignored. Six Sigma proponents claim that the 3.4 DPMO is the long-term process performance after the occurrence of a *sigma shift*. The *sigma shift* is a 1.5 sigma difference from 6 to 4.5 sigma performance. The underlying assumption is that short-term performance (of say 6 sigma) is really 4.5 sigma in the long term as entropy sets in. Sigma shift translates to more defects per million – 1,700 times more. Statistical 6 sigma is not 3.4 DPMO, it is actually 2 DPBO; that is defects per *billion* opportunities, a difference factor of 1,700.

Did you just get a sense of uneasiness? Remember that most companies claiming the use of six sigma for process improvement are not using either of these statistical values; they are merely targeting their processes for measured improvement.

What if performance improved over time, though, in contrast to being subject to entropy? A sigma shift for better would be a 7.5 sigma process. A 7.5 sigma process would have three defects per hundred trillion (3.1 DPhTO [Schofield notation]).

While a 7.5 sigma process seems an unreasonable expectation, at this rate the commercial airline industry would encounter a fatal event every 17,500 years, U.S. highways would incur 23 deaths per year instead of 40,000, and three deaths per annum would be realized from prescription defects instead of 7,000.

But a 7.5 sigma performance is not unreasonable in the computing world. Consider for a moment a teraflop machine that operates at one trillion floating point operations per second. In a mere 100 seconds, three defects would be generated. Within one year, 1,246,080 defects would be generated.

It gets worse. Within the next year (or so) the petaflop machine will be released. A machine operating at that speed could generate over more than one billion defects per year if operating at 7.5 sigma. Do you feel more unease? Do not get prematurely paranoid – a petaflop machine is unlikely to appear on your desktop anytime soon.

Fortunately, hardware performs far more reliably than the sigma levels just described suggest, but that does not apply to software. Software defects cost the U.S. economy almost $60 billion a year [1]. Of course, software defects are not limited to software. Auto companies such as BMW, DaimlerChrysler, Mitsubishi, and Volvo have all experienced software-related product malfunctions (defects) that include engine stalls, wiping interval problems, gauge illumination defects, and transmission gear errors [2]. Software technicians in Panama were charged with murder after 21 patients died from gamma ray overdoses in just 40 months [3]. Sorry, no sigma levels released. And yet, 62 percent of polled organizations lack a software quality assurance group [4].

Practicing statistical *something sigma* is an industry best practice. The Software Engineering Institute's Capability Maturity Model® Integration recognizes the relevance of measurements and analysis by placing it prominently as a Level 2 Process Area in its staged representation. Later in the model, there is the need to identify assignable and common cause variation at maturity Levels 4 and 5, respectively.

So when did statistical notions become ambiguous with words like Lean and Six Sigma? Perhaps organizations should raise an *alert* when the term *six sigma* is used to investigate its contextual alignment with

expectations, visions, and goals. Perhaps, too, the process improvement initiative will have an increased likelihood of success – regardless of what it is called.

Conclusion

Given the abundance of quality improvement and Six Sigma tools available to organizations today, incorporating six sigma measurements might not be that difficult – if the organization chooses to do so. For instance, brainstorming techniques for current state weaknesses could be validated with statistical data (perhaps not to a six sigma threshold, but the introduction of any statistical validation on root cause analysis might provide relevant insight into weaknesses). Root causes listed on cause and effect (Fishbone) diagrams could similarly be validated with statistical data collection. Process flow maps could use the distribution of a statistical sample in assigning hands-on and queue time measurements. Each of these uses of statistics would begin to reintroduce the use of quantitative measures into the Six Sigma movement, perhaps leading to the reemergence of six sigma quality thresholds.

Mark Twain probably was not thinking about Six Sigma when he described the three types of lies as lies, darned lies (paraphrased), and statistics, but his quote seems apropos given how Six Sigma proponents use six sigma today. Six Sigma should be reserved for, well, six sigma performance – a statistical measure for variation. Maybe then quality will translate to fewer product recalls, lower costs will mean that costs are decreased, and six sigma performance will equate to two defects per billion – maybe that is asking too much. Distinguishing between statistically measured performance and measured performance can help assess the true progress of an improvement effort. When applying Six Sigma for process improvement, do not leave out the six sigma.◆

References

1. Zaino, Jennifer. "Behind the Numbers." Information Week. 29 Mar. 2004: 94.
2. Sullivan, Laurie. "Software Quality." Information Week. 15 Mar. 2004: 56.
3. Gage, Debbie, and John McCormick. "We Did Nothing Wrong." Baseline-The Project Management Center, 4 Mar. 2004.
4. Surmacz, Jon. "Why Software Quality Stinks." CIO. 1 Dec. 2003: 28.

About the Author

Joe Schofield is a distinguished member of the technical staff at Sandia National Laboratories. He is a trained Lean Six Sigma Black Belt, chairs the organization's Software Engineering Process Group, is the Software Quality Assurance Group leader, and is accountable for the introduction of the Personal Software Process℠ and Team Software Process℠. He has dozens of publications and conference presentations. Schofield chairs the Management Reporting Committee for the International Function Point Users Group, is active in the local Software Process Improvement Network, and has taught graduate level software engineering classes since 1990.

Sandia National Laboratories
MS 0661
Albuquerque, NM 87185
Phone: (505) 844-7977
Fax: (505) 844 2018
E-mail: jrschof@sandia.gov

Win the Battle, Lose the War

Readers of my BACKTALK columns will already know that I am a Civil War buff, and my recent move to New Mexico brought new sites to explore. I did not realize that the West was the site of many important Civil War battles. In fact, within 60 miles of Albuquerque, the Battle of Glorieta Pass was fought. This little-known battle has been called the *Gettysburg of the West*. It was such an important Northern victory that it effectively signaled the end of Confederate influence in the Southwest.

The Confederates were interested in the West. They wanted recognition by Mexico, and they also wanted the gold and silver that was in Colorado, Nevada, and California. Unfortunately for the Confederates, New Mexico (which fought on the Union side) stood in the way. Texas, a Confederate state, sent forces up the Rio Grande. The Confederates captured Fort Fillmore (near Las Cruces, N.M.), proceeded to win the Battle of Valverde, and advanced north up the Rio Grande. The Confederates eventually occupied Albuquerque and Santa Fe. Their primary objective was to take Fort Union, N.M. – an important Union federal supply center. Fort Union sat directly on the Santa Fe Trail, and is about 25 miles from present-day Las Vegas, N.M. It is about 50 miles northeast of Santa Fe as the crow flies (assuming the crow can fly over pretty rough mountain passes), but 100 miles away via the Santa Fe Trail. To combat the Confederate advance, Union forces from Fort Union, augmented by a regiment of First Colorado Volunteers, advanced south on the Santa Fe Trail towards Glorieta Pass.

The Battle of Glorieta Pass itself was fought from March 26-28, 1862 (the Battle of Gettysburg also took three days). For the first two days of the battle, there was mixed fighting at several locations near Santa Fe with inconclusive results. On March 28, the Confederates advanced toward Fort Union, initially heading southeast towards Glorieta Pass. In what proved to be a very unfortunate decision, the Confederates left all their supplies in a wagon train at Cañoncito, about halfway between Santa Fe and Glorieta Pass. This wagon train was guarded by a single cannon and a handful of noncombatants. This poorly defended supply train (about eighty wagons) contained the entire Confederate reserves of ammunition, baggage, food, forage, horses, mules, and medicines.

The Confederate forces proceeded toward Fort Union and met the advancing Union forces at Pigeon's Rest, slightly east of Glorieta Pass. The Confederates thought their supplies were safe – after all, for Union forces to reach the Confederate supplies, the Union forces would have to go through the Confederates. However, as the battle raged around Pigeon's Ranch, a small group of Union forces were dispatched to find and destroy the Confederate supply train. Since the Union forces consisted of frontiersmen from mining districts near Denver, mountainous terrain did not deter them. The Union dispatchment avoided the Santa Fe Trail, bypassed the Confederate forces, and crossed over 16 miles of mountainous terrain. They then located, attacked, and destroyed the entire Confederate supply wagon train at Cañoncito. The Union troops retraced their route and rejoined the main Union forces after dark (as the battle was ending).

The Confederates went to sleep that night thinking they had won this battle, just as they had won all their previous battles in New Mexico. The Union forces also thought they had won.

Casualties on each side were about the same – about 50 killed and 60 wounded. But it wasn't this relatively indecisive battle that was important. Unbeknownst to the Confederates, the destruction of their supply train checked the advance of the Confederate forces in New Mexico. Just as the Battle of Gettysburg was the *high-water mark of the Confederacy*, the Battle of Glorieta Pass was definitely the *high-water mark of the Confederacy in the Southwest*.

The Confederates' lack of supplies eventually forced them to retreat, backtrack down the Rio Grande, and return to San Antonio. It is now recognized that the Battle of Glorieta Pass effectively stopped a Confederate invasion in the Southwest. The battle signaled the end of a valorous Confederate presence along the Rio Grande in the War of Northern Aggression (I AM a Southerner and a Texas A&M grad – it was either put this in or get tarred and feathered at my next family or class reunion).

What does all this possibly have to do with maturity models? Because simply developing code is like fighting a battle. Winning one battle is not enough. To win the war, you need to be able to fully support all your assets. In the battle for the Southwest along the Rio Grande, the Confederates won most, if not all, of the battles. However, it was the lack of support and assets that cost them the war.

It doesn't really do you any good to deliver code to your users if you cannot provide support for maintenance and updates. You need a process (or maturity model) in place to ensure that you can provide long-time support. Just as losing the supply train spelled the end of the Confederacy along the Rio Grande, problems with life-cycle support can easily spell the end of your development effort.

Maturity models have to support everything you are going to need to eventually win the war, not just the upcoming battle. If it doesn't, then you need to implement whatever process it takes. If you are only worried about the current skirmish, ask yourself this: "Is my lack of configuration management, risk management, or requirements management going to eventually cost me the victory?" Short-term thinking wins battles. Long-term thinking wins wars.

Do you have a maturity model in place? Does it work? If not, then why aren't you fixing it?

Do you want to win a battle, or win the war?

— David A. Cook, Ph.D.
Senior Research Scientist (and Civil War Buff)
The AEgis Technologies Group, Inc.
dcook@aegistg.com

Additional Reading

1. See <www.santafetrailnm.org>, <http://web.archive.org/web/20001002020035>, <www.nmhu.edu/researchsftrail/mapsft1.gif>, <http://web.archive.org/web/20001002020035/>, and <www.nmhu.edu/research/sftrail/mapsft1.gif> for maps and more information on Glorieta Pass.
2. Information in the preceding paragraphs has been taken from an article by Don E. Alberts at <www.tsha.utexas.edu/handbook/online/articles/GG/qfg2.html>, <http://www.tsha.utexas.edu/handbook/online/articles/GG/qfg2.html>, and also from <http://history.sandiego.edu/gen/civilwar/14/glorieta.html> and <http://history.sandiego.edugen/civilwar/14/glorieta.html>. Additional information has been taken from personal trips to New Mexico's Las Cruces, Glorieta Pass, Pecos National Historical Park, and Ft Union National Monument. And no, I am not in the employ of the New Mexico Department of Tourism.

CrossTalk / 309 SMXG/MXDB
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737